

Scalable, Integrated Solutions for Elastic Caching Using IBM WebSphere eXtreme Scale

Easily scalable solutions for elastic
data caching

Integration with IBM WebSphere
Commerce Server

Integration with IBM
WebSphere Portal Server



Priyanka Arora
Deepak Khandelwal
Jonathan Marshall
Abhilash Usha

Redbooks



International Technical Support Organization

**Scalable, Integrated Solutions for Elastic Caching
Using IBM WebSphere eXtreme Scale**

February 2011

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (February 2011)

This edition applies to WebSphere eXtreme Scale V7.1.

© Copyright International Business Machines Corporation 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contact an IBM Software Services Sales Specialist



Start SMALL, Start BIG, ... **JUST START** architectural knowledge, skills, research and development . . . **that's IBM Software Services for WebSphere.**

Our highly skilled consultants make it easy for you to design, build, test and deploy solutions, helping you build a smarter and more efficient business. **Our worldwide network of services specialists wants you to have it all!** Implementation, migration, architecture and design services: IBM Software Services has the right fit for you. We also deliver just-in-time, customized workshops and education tailored for your business needs. You have the knowledge, now reach out to the experts who can help you extend and realize the value.

For a WebSphere services solution that fits your needs, contact an IBM Software Services Sales Specialist:
ibm.com/developerworks/websphere/services/contacts.html

Contents

Contact an IBM Software Services Sales Specialist	iii
Notices	ix
Trademarks	x
Preface	xi
The team who wrote this book	xi
Now you can become a published author, too!	xii
Comments welcome.	xiii
Stay connected to IBM Redbooks	xiii
Chapter 1. Introduction to WebSphere eXtreme Scale	1
1.1 Scalability challenges	2
1.2 Scalability solutions.	4
1.3 WebSphere scalability solutions	6
1.4 WebSphere eXtreme Scale.	6
1.4.1 Highly available and scalable elastic grids	7
1.4.2 Extreme transaction support.	7
1.4.3 Security	7
1.4.4 Runtime JVM support	8
1.4.5 Monitoring techniques.	8
1.4.6 When to use WebSphere eXtreme Scale	8
1.5 IBM WebSphere DataPower XC10 Appliance	12
1.6 Conclusion	14
1.6.1 Cost savings	14
1.6.2 Adapting to growing business needs	14
Chapter 2. Integrating WebSphere eXtreme Scale with other middleware	15
2.1 WebSphere Business Events and WebSphere eXtreme Scale	16
2.1.1 Challenges	16
2.1.2 WebSphere eXtreme Scale benefits.	18
2.1.3 For more information.	20
2.2 Rational Jazz based products and WebSphere eXtreme Scale	21
2.2.1 Challenges	21
2.2.2 WebSphere eXtreme Scale benefits.	21
2.2.3 Where to find more information.	23
2.3 WebSphere Commerce and WebSphere eXtreme Scale	23
2.3.1 Challenges	23
2.3.2 WebSphere eXtreme Scale benefits.	25
2.3.3 When to use WebSphere eXtreme Scale	26
2.3.4 Where to find more information.	27
2.4 WebSphere Portal Server and WebSphere eXtreme Scale	27
2.4.1 Challenges	27
2.4.2 Integration architecture	28
2.4.3 WebSphere eXtreme Scale benefits.	29
2.4.4 Where to find more information.	30
Chapter 3. WebSphere eXtreme Scale Architecture	31
3.1 WebSphere eXtreme Scale architecture.	32

3.1.1 User view	32
3.2 Grid client and servers	35
3.3 Catalog service	36
3.4 Shard placement	37
3.5 Zone support	39
3.5.1 Zone-based routing	40
3.6 Scalability sizing considerations	40
3.6.1 Heap size and the number of JVMs	40
3.6.2 Number of grids	41
3.6.3 Catalog servers	41
3.6.4 Sizing for growth	41
3.7 Common topologies	41
3.7.1 Offloading data from server JVMs	41
3.7.2 Collocating cache with server JVMs	43
3.7.3 High availability and multiple data center topologies	45
3.8 Monitoring eXtreme Scale	47
3.8.1 Using the eXtreme Scale web console	47
3.8.2 Monitoring with xsadmin	49
3.8.3 Monitoring with the Tivoli Performance Viewer	50
3.8.4 Monitoring MBeans statistics with wsadmin	53
3.8.5 Monitoring MBean statistics in JConsole	55
3.8.6 Monitoring with vendor tools	55
Chapter 4. Installing WebSphere eXtreme Scale	57
4.1 Installing with an existing WebSphere Application Server product	58
4.1.1 Installing WebSphere eXtreme Scale	58
4.1.2 Creating, starting and stopping catalog servers and containers	63
4.2 Installing stand-alone WebSphere eXtreme Scale	63
4.2.1 Installing a stand-alone environment	64
4.2.2 Starting a catalog server on a stand-alone installation	66
4.2.3 Stopping a catalog server	66
4.2.4 Starting grid containers on a stand-alone installation	66
4.2.5 Stopping the grid containers	66
4.3 Applying maintenance	67
Chapter 5. Integrating WebSphere Portal with WebSphere eXtreme Scale	69
5.1 When to use eXtreme Scale with Portal	70
5.2 Integration configuration and environment	70
5.2.1 Prerequisites and requirements	70
5.2.2 Sample environment	71
5.2.3 Summary of the configuration	71
5.2.4 Custom portlets	72
5.3 Integration details	73
5.3.1 Installing WebSphere eXtreme Scale for integration with Portal	73
5.3.2 Configuring eXtreme Scale	74
5.3.3 Configuring WebSphere Portal Server	85
5.3.4 Validating the eXtreme Scale integration	90
5.4 Monitoring	92
5.4.1 Monitoring eXtreme Scale with the xsadmin utility	93
5.4.2 Monitoring eXtreme Scale with the Tivoli Performance Viewer	95
5.5 Guidelines and best practices	95
5.5.1 Deployment	95
5.5.2 Application code	96

5.5.3 JVM tuning	97
Chapter 6. Integrating WebSphere Commerce with WebSphere eXtreme Scale	99
6.1 Why use eXtreme Scale with WebSphere Commerce?	100
6.2 Overview of caching differences with eXtreme Scale	100
6.2.1 Dynamic cache service	100
6.2.2 What to expect when using WebSphere eXtreme Scale	102
6.3 How to integrate eXtreme Scale with WebSphere Commerce	107
6.3.1 Sample environment	107
6.3.2 Summary of configuration	107
6.3.3 WebSphere eXtreme Scale prerequisites	108
6.3.4 Installing WebSphere eXtreme Scale for integration with Commerce	109
6.3.5 Sample environment configuration	110
6.4 Monitoring considerations	132
6.4.1 The Cache Monitor	133
6.4.2 The Tivoli Performance Viewer	134
6.4.3 xsadmin	136
6.4.4 Further monitoring options	136
6.5 Sizing guidance	137
Related publications	141
IBM Redbooks	141
Online resources	141
Help from IBM	143

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.


Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DataPower®
developerWorks®
IBM®
Jazz™
Lotus®

MVS™
Quickr™
Rational Team Concert™
Rational®
Redbooks®

Redpaper™
Redbooks (logo) ®
Tivoli®
WebSphere®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM® WebSphere eXtreme Scale provides a powerful, elastic, high-performance solution for scalability issues through caching and grid technology. This IBM Redbooks® publication shows architects and IT personnel how to leverage the power of WebSphere eXtreme Scale technology to enhance data caching performance in their enterprise networks.

This book discusses the scalability challenges and solutions facing today's dynamic business and IT environments. Topics discussed include existing scalability solutions, how WebSphere eXtreme Scale can be integrated into these solutions, and best practices for using WebSphere eXtreme Scale in different environments, including application data caching and database caching. Also included is an in-depth discussion of the WebSphere eXtreme Scale infrastructure, such as grid clients and servers, the grid catalog service, zone support, and scalability sizing considerations.

This book focuses on the challenges and benefits of integrating WebSphere eXtreme Scale with other middleware products, including WebSphere® Business Events, WebSphere Commerce, WebSphere Portal, and Rational® Jazz™-based products. Detailed procedures for integrating, configuring, and monitoring WebSphere eXtreme Scale in WebSphere Portal and WebSphere Commerce environments are provided.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



Priyanka Arora is a Solution Performance Analyst with the High Performance On Demand Solutions team in India. She has over three years of experience with technical analysis, tuning, troubleshooting, tooling, and load test execution of end-to-end performance engagements for several IBM Telecom & Public Sector customers. She has also worked through various stages of SDLC on Portal Solution development. She is an MSc Tech in Information Systems from Birla Institute of Technology and Science in India and a certified WebSphere Application Server Administrator.



Deepak Khandelwal is an Application Integration and Middleware IT Specialist at IBM Software Group in India. He has been working with IBM since 2007. He has a Master's degree in Information Technology from Indian Institute of Technology, Roorkee. His areas of expertise include WebSphere Application Server, WebSphere eXtreme Scale, Lotus® Connections and Lotus Quickr™. He has considerable experience on Lotus, WebSphere, and Java/J2EE technologies, and is certified on all of them.



Jonathan Marshall is a Consulting IT Specialist working in the UK as a WebSphere Technical Professional. He has 10 years of experience at IBM with WebSphere Application Server and related products. His areas of expertise include developing for and the administration of WebSphere Application Server, WebSphere eXtreme Scale, and WebSphere Virtual Enterprise. He has previously written developerWorks® articles including one on WebSphere eXtreme Scale. He holds a degree in Computer Science from the University of Warwick.



Abhilash Usha is a Software Specialist working for IBM Software Labs, India. He has over 6 years of experience in the IT Industry. His areas of expertise include design and development of J2EE applications and Websphere Application Server Infrastructure. Currently he works on WebSphere eXtreme Scale projects. He holds a Bachelor of Engineering degree in Information Technology from Bharathiyar University, India.

This project was led by a specialist from the International Technical Support Organization:

Carla Sadtler is a Consulting IT Specialist at the ITSO, Raleigh Center. She writes extensively about WebSphere products and solutions. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative, supporting MVS™ customers. She holds a degree in Mathematics from the University of North Carolina at Greensboro.

Thanks to the following people for their contributions to this project:

Mark Blondell
Steve Branda
Joshua Dettinger
Rustan Eklund
Jeffrey Garratt
Nitin Gaur
Jim Holland
Rohit Kelapure
Jim Krueger
Joseph Mayo
IBM US

Mikhail Genkin
IBM Canada

Dan McGinnes
IBM UK

Stephen Smith
International Technical Support Organization, Raleigh Center

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your

network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Introduction to WebSphere eXtreme Scale

This chapter describes how WebSphere eXtreme Scale addresses the scalability and performance challenges that exist in today's dynamic business and IT environments. An introduction to the key features of WebSphere eXtreme Scale and the IBM WebSphere DataPower XC10 Appliance is also provided.

This chapter includes the following sections:

- ▶ “Scalability challenges” on page 2
- ▶ “Scalability solutions” on page 4
- ▶ “WebSphere scalability solutions” on page 6
- ▶ “WebSphere eXtreme Scale” on page 6
- ▶ “IBM WebSphere DataPower XC10 Appliance” on page 12
- ▶ “Conclusion” on page 14

1.1 Scalability challenges

Scalability is the ability of a system to handle increasing load in a graceful manner. This allows a system to be extended easily. For example, in a system that has linear scaling capabilities, doubling the CPU capacity also doubles the maximum throughput that the system can handle. In general, there are two ways an IT system can be scaled:

- Horizontally, by adding additional hosts to a tier. This is also called *scale out*.
- Vertically, by enlarging the capabilities of a single system. For example, adding CPUs. This is also called *scale up*.

Consider a classical three tier application shown in Figure 1-1. The application server tier is both scaled out by having three hosts, and scaled up by having three application servers on each host. The database tier is scaled up by using a single powerful machine with many CPUs. The database tier is scaled out by having a shadow database using log shipping capability to support reports, analysis, and so on.

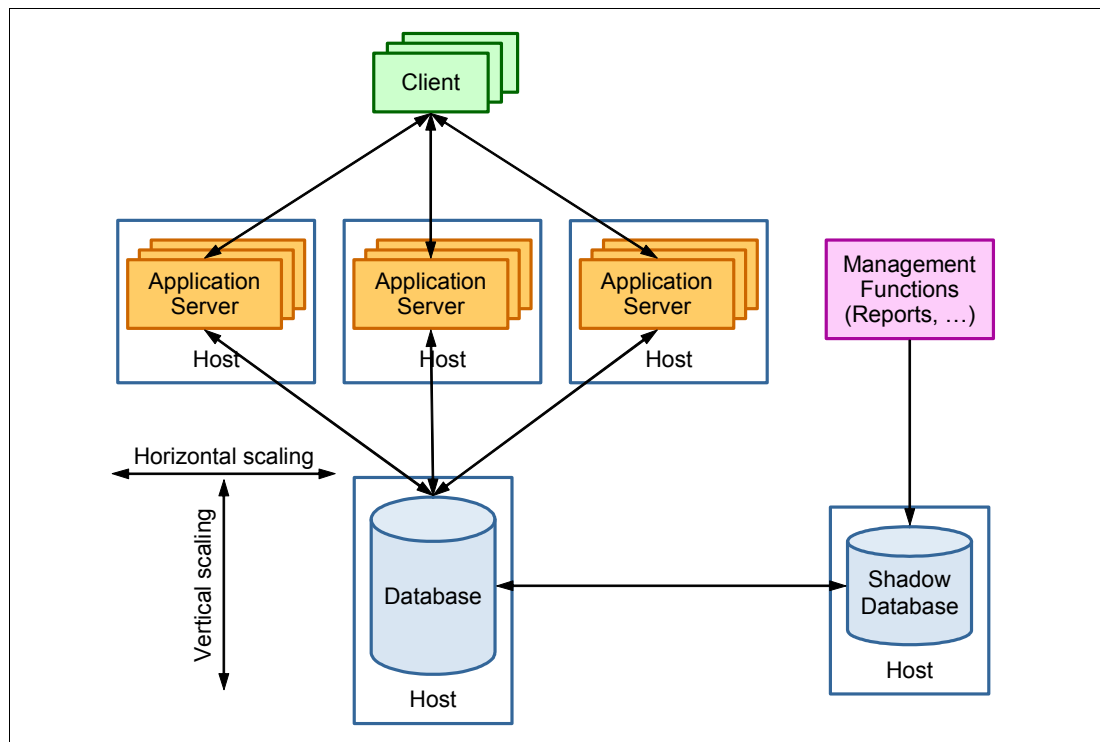


Figure 1-1 Scaling options in a traditional three tier application

Scaling is easy and effective as long as all of the resources involved can cope with the increased load. Eventually a resource will reach its maximum throughput, thereby limiting the overall throughput of a system. This point is called the *saturation point* and the limiting resource is called a *bottleneck resource*.

Figure 1-2 on page 3 shows the correlation between load and throughput that can typically be measured for an application.

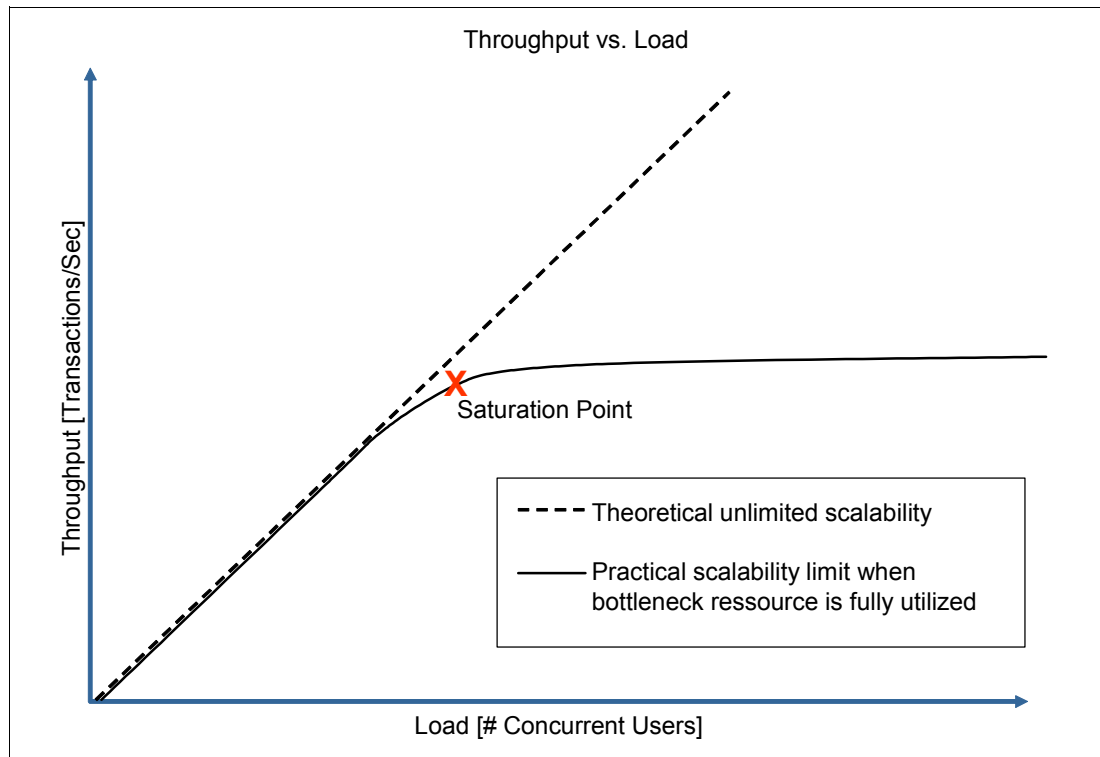


Figure 1-2 Correlation between throughput and load showing scalability limits

In a well-crafted application the database is usually the bottleneck resource (custom non-SQL databases are an exception to this rule). This is because application servers can be effectively scaled horizontally to provide additional memory and processing cycles to service requests. Because the primary resource is the database, a portion of this additional load is passed on to the database.

When the load on the database increases, the usual response is to scale up the database server also. Eventually, either due to practical, financial, or physical limits, the database server is unable to continue to scale up. The progressive approach is to scale out by adding additional database servers and using a high speed connection between them to provide a cluster of database servers. This approach is viable, but poses additional challenges in keeping the database servers synchronized.

Databases must be kept synchronized for data integrity and crash recovery. For example, consider two concurrent transactions that modify the same row in the database. When these transactions are executed by separate database servers, communication is required to ensure the atomic, consistent, isolated, and durable (ACID) attributes of database transaction are preserved. This communication can grow exponentially as the number of database servers increases, which ultimately limits the scalability of the database. In fact, although application server clusters with more than 100 hosts can be easily found, a database server cluster with more than four members is hard to find.

The scalability challenge is to provide scalable access to large amounts of data. In almost all application scenarios, scalability is treated as a competitive advantage. It directly impacts the business applications and the business unit that owns the applications. Applications that are scalable can easily accommodate growth and aid the business functions in analysis and business development. You want to be able to scale your product with predictable costs without requiring a redeployment of an application or topology.

1.2 Scalability solutions

A typical approach to resolving performance and therefore scalability problems is to implement caching of the data. The cache is a copy of frequently accessed data that is held in memory to reduce the access time to the data. This has the effect of placing the data closer to the application, which improves performance and throughput. It also reduces the number of requests to the database, reducing that resource's likelihood of being a bottleneck.

A cache simply extends the storage capability. It can be considered to act as a shock absorber to the database. As shown in Figure 1-3, the cache sits between the application and the database to reduce the load on the database.

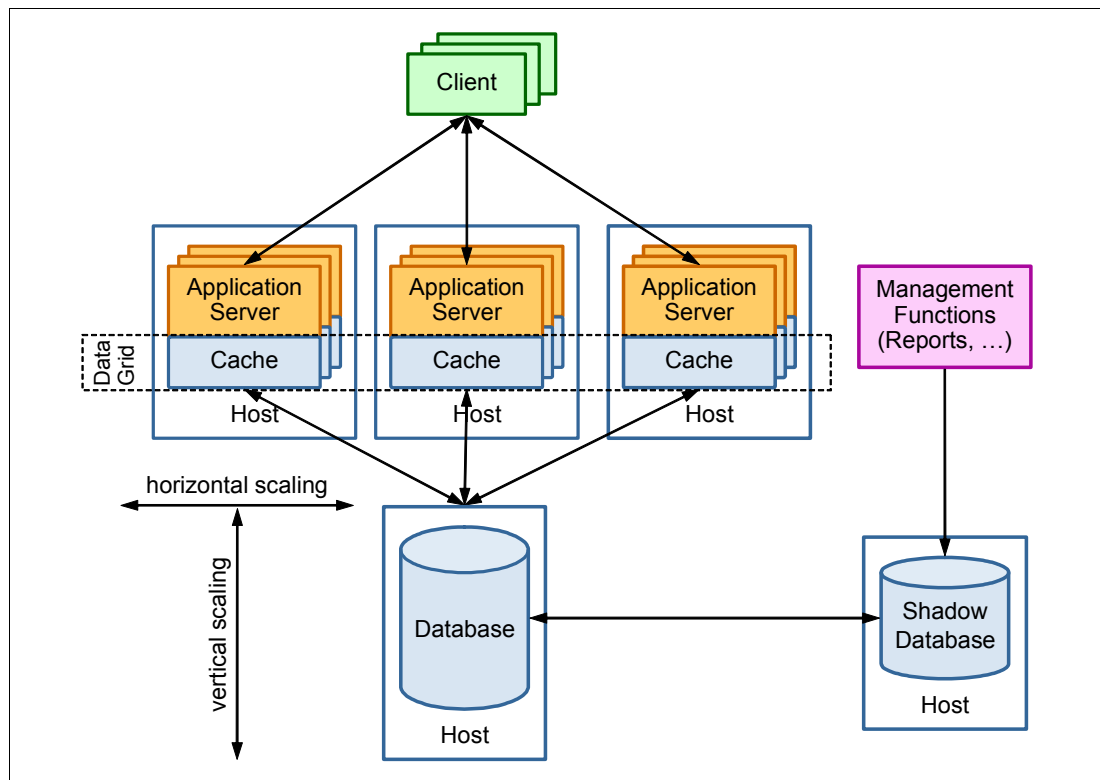


Figure 1-3 Introduce caching as a response to the scalability challenge

Although a cache can reduce the load on the database, the same data might be cached in several servers simultaneously. Things become complicated when one copy of the data is changed, because all cached copies need to be either invalidated or updated.

For very large caching solutions, *data grids* are often used. In general terms, a grid is several loosely-coupled and heterogeneous computers that act together to perform large tasks. To accomplish this, a grid needs to be highly scalable. There are several forms of grids, depending on the task at hand.

The data grid can be located in the same Java™ Virtual Machine (JVM) as the applications (co-located) as shown in Figure 1-3. This arrangement provides the fastest access to the data. However, as the data grid grows, it might not scale effectively because the application and grid share the same address space.

Another approach is to perform local caching in the application server tier (near cache) and to have the data grid cache implementation on a separate elastic tier where the servers are

primarily only responsible for hosting the grid data. This architecture provides additional flexibility in designing the application topology and in the manner of client access (direct to the grid, or through the application server layer), and for the scalability and availability of the grid data as shown in Figure 1-4.

Note: Near cache is not currently supported with the dynamic cache service in WebSphere Application Server.

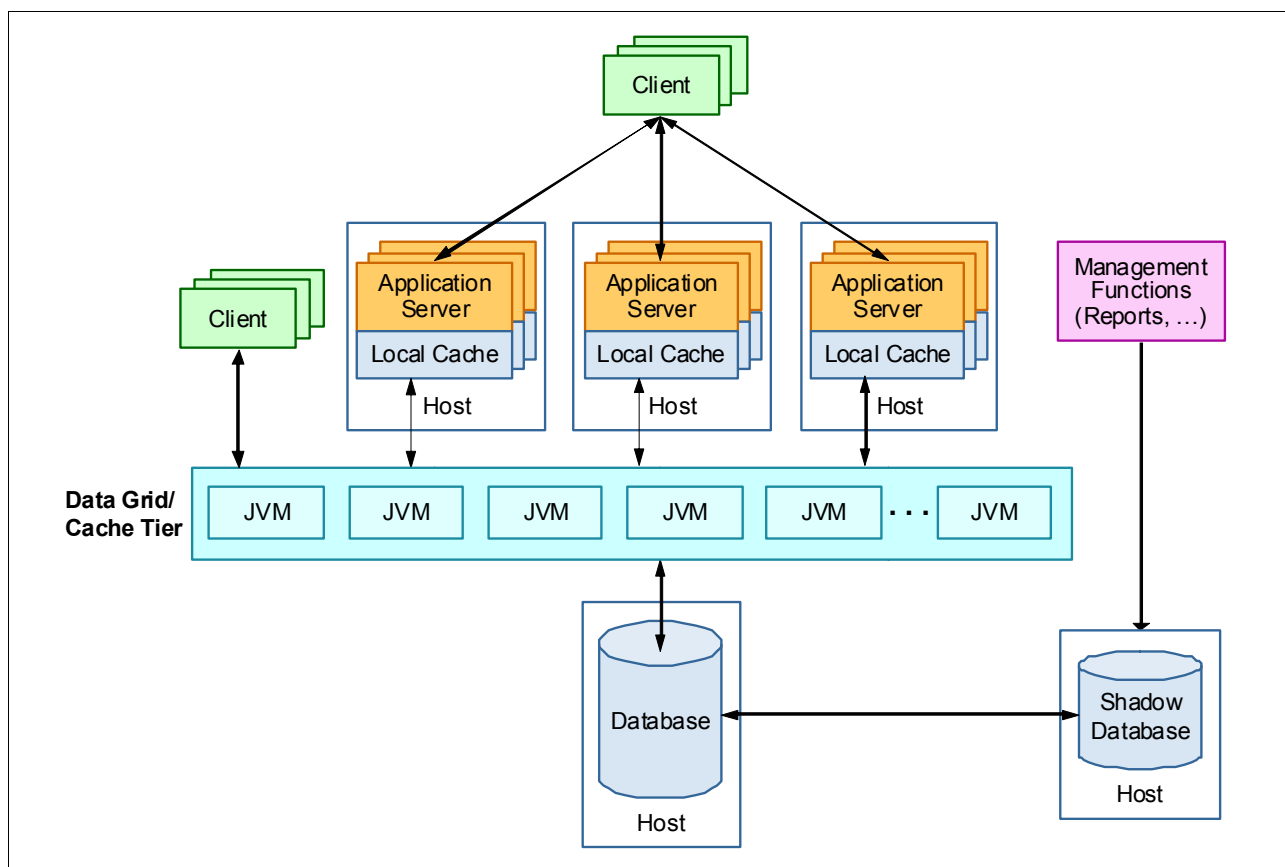


Figure 1-4 Separating the data grid to another tier

As the use of the data grid technology increases, it is possible that the cache or data grid becomes the primary data access point. In this case, the database is used by the grid as a long term persistent data store. If the database is unavailable, the applications are still able to complete transactions against the grid so there is no loss of service to the application. The grid can push those updates to the database when it again becomes available. The database might also be required in application scenarios where logging or auditing of transactions is required for compliance or by regulations. In specialized use cases, such as transitory session state data, there might be no need to store the data from the grid into a database. Because the grid contains all of the information, data intensive computing tasks can then be moved into the grid and executed in parallel.

1.3 WebSphere scalability solutions

Both the IBM WebSphere DataPower XC10 Appliance and WebSphere eXtreme Scale software address scalability issues by introducing elastic data grids into your solution. The data grids hold cached data in a non-SQL structure. Both the software and the appliance allow you to quickly and easily scale out your transaction volumes with minimal invasive changes to your application and architecture. This will also reduce number of read and write operations to the database, resulting in less time required for resource intensive calls. Both solutions are easy to access and cost effective as your business needs grow.

Figure 1-5 shows how the WebSphere scalability solutions fit into the web application topology.

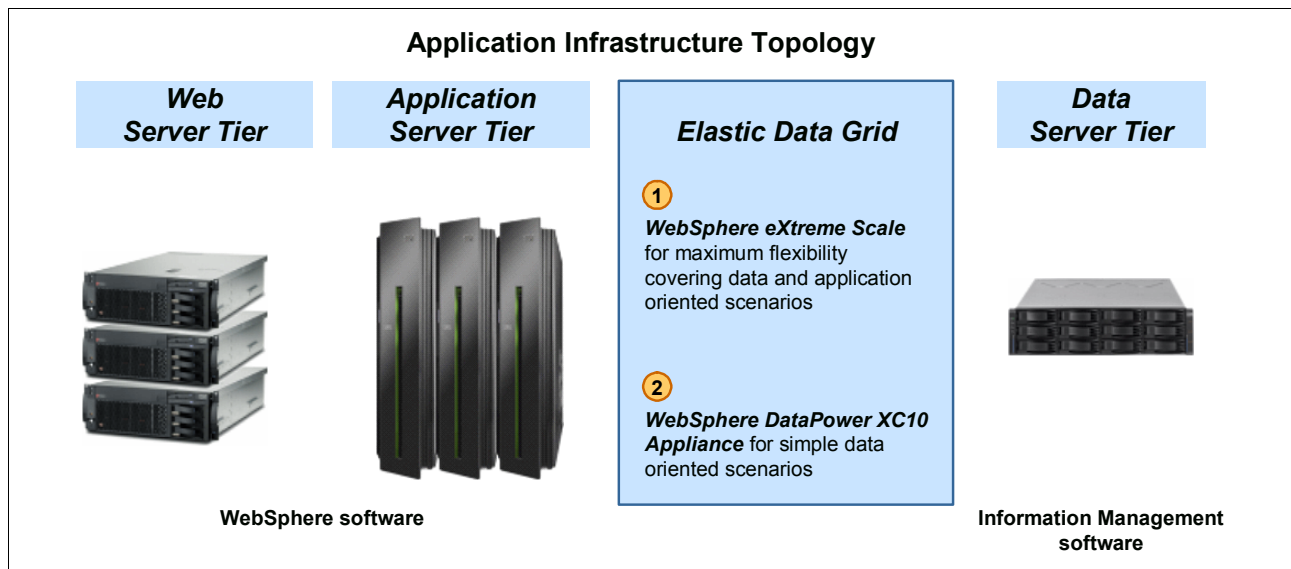


Figure 1-5 IBM WebSphere DataPower XC10 Appliance Infrastructure Topology

Although both the XC10 and WebSphere eXtreme Scale offer scalability and integration features, this book will focus on the solutions available today for integration of WebSphere eXtreme Scale with existing products.

1.4 WebSphere eXtreme Scale

WebSphere eXtreme Scale provides an extensible framework to simplify the caching of data used by an application. It can be used to build a highly scalable, fault tolerant data grid with nearly unlimited horizontal scaling capabilities.

WebSphere eXtreme Scale creates infrastructure with the ability to deal with extreme levels of data processing and performance. When the data and resulting transactions experience incremental or exponential growth, the business performance does not suffer because the grid is easily extended by adding additional capacity in the form of Java virtual machines and hardware.

The key features of WebSphere eXtreme Scale include:

- ▶ A highly available elastic and scalable grid
- ▶ Extreme transaction support
- ▶ Security

- ▶ Easy integration into existing solutions
- ▶ Support for JSE, JEE, ADO.NET data services, and REST capable client applications
- ▶ Monitoring

1.4.1 Highly available and scalable elastic grids

WebSphere eXtreme Scale supports a dynamic grid infrastructure with support for substantial scale outs. It is designed to scale to thousands of server instances. This is accomplished by splitting large amounts of data into manageable chunks and distributing them across the grid.

Each of the server instances that host grid data does so in a grid container. Communication between containers in a grid can be a crucial limiting factor for scalability. This is why WebSphere eXtreme Scale grid containers have been designed to require minimal communication with each other, allowing large linear scale outs. Communication between grid containers is kept to a minimum, occurring only for availability management and data replication purposes.

When the data in the grid must be highly available, the grid can be configured for replication so that in the event of a failure no loss of critical data occurs. The grid data is kept highly available by having multiple instances of the data stored on separate servers, and even in separate locations to ensure recovery in a disaster scenario. This capability is defined when the grid is created.

1.4.2 Extreme transaction support

WebSphere eXtreme Scale has built-in transaction support for all changes made to the cached data. Changes are committed or rolled back in an atomic way. WebSphere eXtreme Scale augments the database and acts as an intermediary between the application and database. It can also be the system of record for applications when no database or other data store is used. Transaction processing ensures that multiple individual operations that work in tandem are treated as a single unit of work. If even one individual operation fails, the entire transaction fails.

As with other persistent store mechanisms, WebSphere eXtreme Scale uses transactions for the following reasons:

- ▶ To apply multiple changes as an atomic unit at commit time
- ▶ To ensure consistency of all cached data
- ▶ To isolate a thread from concurrent changes
- ▶ To act as the unit of replication to make the changes durable.
- ▶ To implement a life cycle for locks on changes
- ▶ To combine multiple changes to reduce the number of remote invocations

1.4.3 Security

If the data that is stored in the cache is of a sensitive nature, fine-grained control over client access to data can be enforced. WebSphere eXtreme Scale applications can enable security features and integrate with external security providers. WebSphere eXtreme Scale includes the following security features:

- ▶ Authentication

Authentication provides the ability to authenticate the identity of the client of the grid. This is done with credential information between the client and the grid server.

- ▶ Authorization

Authorization provides allows you to grant access only to authenticated clients. The authorization includes operations such as reading, querying, and modifying the data in the grid.

- ▶ Transport security

Transport security ensures secure communications between the remote clients and grid servers, and between the servers.

1.4.4 Runtime JVM support

WebSphere eXtreme Scale does not require WebSphere Application Server as a runtime application, although additional functionality for monitoring and maintaining the grid is available with that application. WebSphere eXtreme Scale can use any native JSE (1.4+) or JEE application server environment. Not all features are available with JSE 1.4. Additional capabilities for annotations and JPA support is available with JSE 1.5 or later.

It is also possible to access the data stored in the grid from an ADO.NET Data Services client using the Representational State Transfer (REST) API Data Services Framework. REST is available with WebSphere eXtreme Scale 7.0.0.0 cumulative fix 2 or later.

1.4.5 Monitoring techniques

Monitoring the availability and performance of enterprise data is critical for maintaining the integrity of the application infrastructure. Monitoring of WebSphere eXtreme Scale grids can be done through the following widely used techniques:

- ▶ The **xsadmin** utility can be used to view the grid usage. This utility is included with the standard installation.
- ▶ The WebSphere eXtreme Scale web console provides a GUI view and usage pattern of the grid. The web console is included with the stand-alone installation of WebSphere eXtreme Scale.
- ▶ JConsole can also be leveraged to view the grid statistics. JConsole is not supported with WebSphere Application Server. More information about monitoring can be found in Section 3.8.5, “Monitoring MBean statistics in JConsole” on page 55.
- ▶ Custom MBean Scripts can be written to monitor the cache. A sample custom script that can be used to check the cache usage is included in Section 3.8.3, “Monitoring with the Tivoli Performance Viewer” on page 50.
- ▶ The Tivoli® Performance Viewer (TPV) shows grid statistics when you enable the Object Grid Related PMI statistics.
- ▶ Third party monitoring products such as CA Wily Introscope, and Hyperic HQ can be used with WebSphere eXtreme Scale. IBM Tivoli Monitoring agent and Hyperic HQ gather data from WebSphere eXtreme Scale server side MBeans. CA Wily Introscope uses byte code instrumentation to display data.

1.4.6 When to use WebSphere eXtreme Scale

WebSphere eXtreme Scale provides a data grid to store data. How you access that grid, and the type of data you store in it can vary depending on the application. A detailed systems and design analysis is required to determine if, when, and how you use WebSphere eXtreme Scale. This is an important exercise for devising a road map for adoption of any new technology. The decision tree shown in Figure 1-6 on page 9 can help in the evaluation.

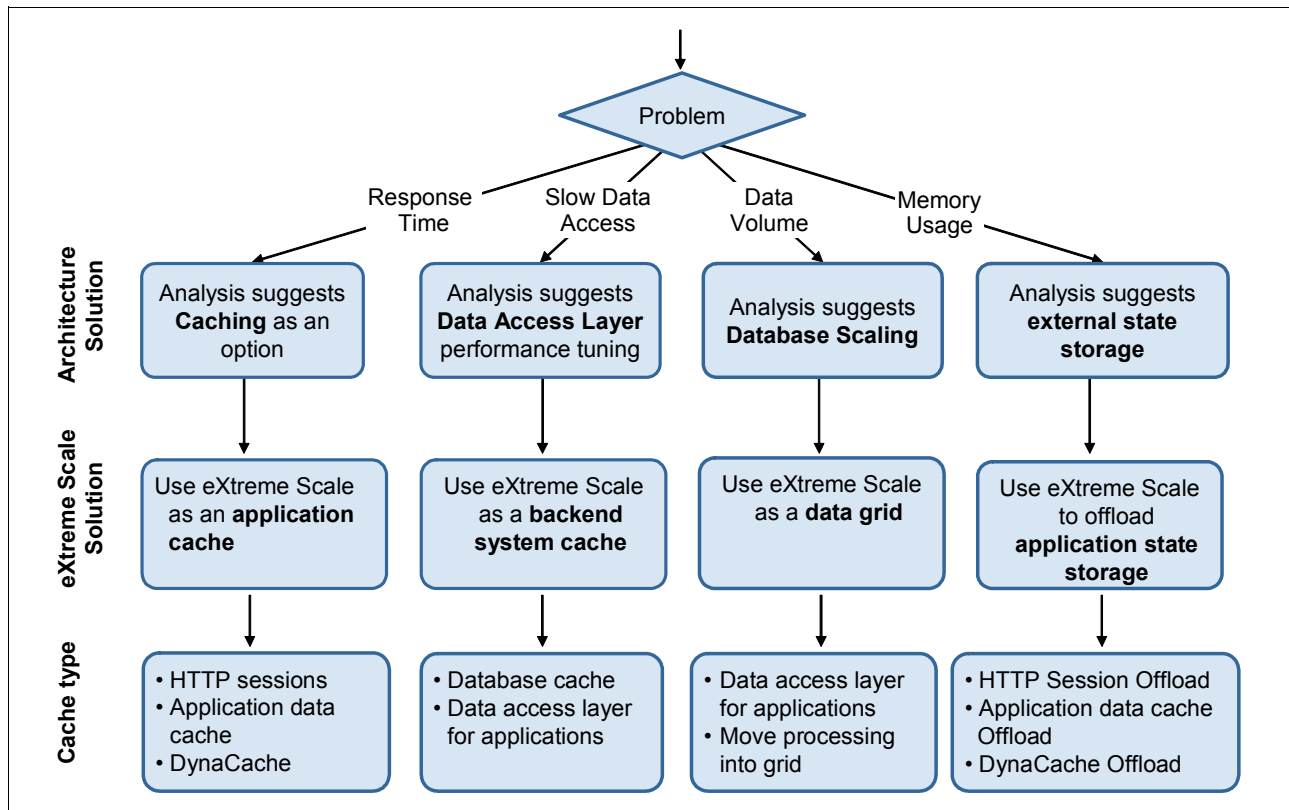


Figure 1-6 Decision tree for adopting WebSphere eXtreme Scale

The decision tree addresses the issues of inadequate response time, slow data access, data volume issues that affect performance, and memory usage problems. Based on analysis of similar problems, the first tier in the tree suggests the type of solution appropriate at an architectural level. The second tier in the tree suggests how WebSphere eXtreme Scale can be used to implement the solution. The third tier provides information about the specific types of caching that might prove useful in resolving the problem.

The problem categories shown in the decision tree are interrelated, in that high database load will lead to slow data access, which by itself is a cause of slow application response times. Removing one scalability bottleneck by introducing caching or a more advanced data grid solution will often reveal another bottleneck, so it might take a few iterations and a few uses of WebSphere eXtreme Scale before the application meets its scalability targets.

The problems shown in the decision tree invite a number of incremental solutions, such as tuning database settings for greater performance, optimizing queries, and scaling up database servers. These do not address the greater scalability limitations of the application, but rather move it towards the limits of the current architecture. The effectiveness of these solutions depends on the inefficiency of the application and the availability of more powerful server equipment, both of which can quickly reach the point of diminishing returns.

The following sections provide more information about the options found in the third tier of the decision tree in Figure 1-6.

HTTP sessions

Many web applications use the HTTP session state management features provided by Java Enterprise Edition (JEE) application servers. Session state management allows the application to maintain state for the period of a user's interaction with the application. An

example of this is a shopping cart, where information about intended purchases is stored until the shopping session is completed.

To support high-availability and failover, the state information must be available to all application server JVMs. Application servers typically achieve this by storing the state information in a shared database, or by performing memory-to-memory replication between JVMs. When HTTP session state is stored in a shared database, scalability is limited by the database server. The disk I/O operations have a significant impact on application performance. When the transaction rate exceeds the capacity of the database server, the database server must be scaled up.

When HTTP session state is replicated in memory between application servers, the limits to scalability vary depending on the replication scheme. Commonly, a simple scheme is used in which each JVM holds a copy of all user session data, so the total amount of state information cannot exceed the available memory of any single JVM. Memory-to-memory replication schemes often trade consistency for performance, meaning that in cases of application server failure or user sessions being routed to multiple application servers, the user experience might be inconsistent and confusing.

WebSphere eXtreme Scale is an ideal platform to store HTTP session data. An eXtreme Scale approach replaces the existing HTTP session state management facilities of the application server, placing the session data in a data grid. WebSphere eXtreme Scale fetches user session information from the grid and writes changes back to the grid as necessary. Because the HTTP session data is transient in nature, it does not need to be backed up to disk and can be contained completely in a highly-available replicated grid. The grid is not constrained to any one application server product or to any particular management unit, such as WebSphere Application Server cells. User sessions can be shared between any set of application servers - even across data centers in the case of a failover scenario. This allows for a more reliable and fault-tolerant user session state.

There are two topologies you can use with HTTP session offloading:

- Collocated HTTP sessions

Using this method, HTTP session grids coexist with the application in the JVMs. However, the session data is partitioned and distributed, making it highly scalable, fault tolerant, and much faster.

- HTTP session offloading

Using this method, HTTP session data grids are offloaded from the JVM running the application and placed into external eXtreme Scale containers. This reduction of session data held in the application JVM can increase the efficiency of the application.

An example of using WebSphere eXtreme Scale HTTP session support with WebSphere Portal Server for customer portlets can be seen in Chapter 5, “Integrating WebSphere Portal with WebSphere eXtreme Scale” on page 69.

Dynamic cache

Many web-based applications use dynamic page generation techniques, like JavaServer Pages (JSPs) for pages that contain data that rarely changes, such as product details or information about corporate policies. Application servers generally provide an in-memory cache to store the generated output the first time the page is rendered to save the processing work and back-end system load for subsequent requests.

WebSphere Application Server’s dynamic cache service stores the output of servlets, JSPs, and custom application objects. The dynamic cache service can be configured to off-load cached objects to disk, or to replicate cached objects across a cluster of application servers.

When the disk off-load option is enabled, the dynamic cache service extends its in-memory cache by writing cached objects to disk when the in-memory cache is full. If the working set of the application does not fit within the in-memory cache, disk access time becomes an important factor in application performance. Faster disk technologies, such as Storage Area Networks (SANs), are often used to improve performance.

When memory-to-memory cache replication is used, the size of the cache is limited to the available memory of any single JVM hosting the application. If the working set of the application outgrows the cache, the application will perform redundant work to re-create objects that have been evicted from the cache.

WebSphere eXtreme Scale provides a drop-in replacement for the dynamic cache service that stores cached objects in a grid, allowing the size of the cache to expand to the sum of all available memory in JVMs in the grid. The application's dynamic cache service policy remains in effect, so the carefully defined dependency and invalidation rules continue to function.

One benefit is that a newly started application server no longer has to warm up its cache before reaching peak performance. The cached data is already available to the application server from the grid. As a result, the application's performance is more reliable, and scaling out the application does not result in load spikes at the back-end systems.

Memory usage for caching can be reduced by as much as 70 percent compared to per-server disk off-load caches, owing to the reduced redundancy of a single shared cache. Additionally, the eXtreme Scale dynamic cache service provider can compress the contents of the cache to save even more memory.

An example of using WebSphere eXtreme Scale as a dynamic cache provider can be seen in Chapter 6, "Integrating WebSphere Commerce with WebSphere eXtreme Scale" on page 99.

Application data cache

WebSphere eXtreme Scale can be used to cache internal application data that is not directly represented in a database or other back-end system. This expands the amount of such data that can be cached by the application and improves the cache hit rate by sharing the cache across application servers and, potentially, between related applications.

This type of caching must be implemented within the application, requiring development effort. For more information about using eXtreme Scale as an application data cache, see *User's Guide to WebSphere eXtreme Scale*, SG24-7683.

Database cache

The back-end database is the scalability bottleneck for many database-centric applications. As the data volume and transaction rate expand, the application will eventually reach a point where the database server cannot be scaled up any further.

WebSphere eXtreme Scale can be used as a scalable, shared, coherent cache for databases and other data sources, reducing the load on the back-end systems and improving the performance of the application. It offers drop-in caching solutions for the OpenJPA and Hibernate database persistence layers, meaning that only configuration changes are required for certain applications.

For more information, go to the Information Center for the L2 cache Plugin Support at the following address:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r0/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/cxscacheint.html>

Using the grid as a data access layer

Using WebSphere eXtreme Scale as a data access layer allows access to its querying and indexing capabilities, which can reduce the load on back-end systems. This approach involves replacing the application's existing data access layer using the WebSphere eXtreme Scale programming interfaces. In this case, a grid acts as a cache for the database. The database remains the system of record, but all data access from the application goes through the cache grid. The application uses the WebSphere eXtreme Scale programming interfaces to access the data.

Using this method, the application can use the JPA-based database loader provided with WebSphere eXtreme Scale or a custom loader to retrieve and update grid entries from the back-end data store.

For more information about using eXtreme Scale as a data access layer, see *User's Guide to WebSphere eXtreme Scale*, SG24-7683. Another article of interest is *IBM Extreme Transaction Processing (XTP) Patterns: Leveraging WebSphere eXtreme Scale as an in-line database buffer* that discusses how to use eXtreme Scale as a write-behind JPA cache located at the following address:

http://www.ibm.com/developerworks/websphere/library/techarticles/0906_vuong/0906_vuong.html

Moving application processing into the grid

Beyond caching and data access scenarios, WebSphere eXtreme Scale offers a programming model for processing data sets in parallel within the grid containers that hold the data. Inserting a fully populated, partitioned, in-memory cache into the application architecture allows you to process the entire data set in parallel at local memory access speeds. Instead of fetching data from the back-end data store and processing it in series, the application can process the data in parallel across multiple grid containers and combine the results together.

This approach requires significant application design effort to identify the operations to perform within the grid, and to implement those operations using the WebSphere eXtreme Scale programming interfaces.

This approach enables parallel processing of large volumes of data in ways that were not practical without the grid. It might even be possible to run processes online that previously could only be run as batch jobs.

WebSphere Business Events uses eXtreme Scale to handle complex event processing. An overview of this scenario can be found in Chapter 2, "Integrating WebSphere eXtreme Scale with other middleware" on page 15.

1.5 IBM WebSphere DataPower XC10 Appliance

The IBM WebSphere DataPower XC10 Appliance, shown in Figure 1-7 on page 13, is a purpose-built, easy-to-use appliance designed for simplified deployment at the caching tier of the enterprise application infrastructure.



Figure 1-7 IBM WebSphere DataPower XC10 Appliance

The DataPower XC10 Appliance is designed for rapid, “drop-in” use in conjunction with WebSphere Application Server and other WebSphere family products. It is based on the DataPower® 9004 platform. Because the required data can be stored on the appliance in an in-memory grid, application caching functions can be performed extremely fast and can scale with consistent performance.

With the DataPower XC10 Appliance, you can experience the following benefits:

- ▶ **Easy scale-out**

The DataPower XC10 Appliance contains a 160 GB elastic data cache in a high density, low footprint that allows you to save time, money, and rack space for business critical applications. The DataPower XC10 appliance stores data on solid-state disk (SSD). To add more memory to your data grid, you can add another appliance to your configuration, creating a collective of appliances that host your data. It provides linear scaling according to the application without application downtime.

- ▶ **Drop-in use without code changes**

The DataPower XC10 Appliance can be used without extensive changes to the code in existing applications. There are no application code changes required to use the DataPower XC10 Appliance with WebSphere Application Server for session management or as a provider for the dynamic cache service.

The DataPower XC10 Appliance can be used in the following drop-in scenarios:

- Session management for HTTP requests
- WebSphere Application Server dynamic cache support
- Simple cache using eXtreme Scale APIs

- ▶ **Fault tolerance**

The data in the data grids is automatically replicated, which lowers the risk of data loss. The DataPower XC10 Appliance provides the ability to group multiple appliances together into a collective for availability, scalability, and management purposes. The DataPower XC10 Appliance provides self healing by detecting failures automatically.

- ▶ **Flexible and simple user management**

With the web console, you can easily manage users and user groups for your appliance. You can also use the web console to create, manage, and monitor your data grids. Native interfaces are also provided for administration and monitoring.

1.6 Conclusion

WebSphere eXtreme Scale addresses two major concerns for scalability: cost savings and adaptation to growing business needs.

1.6.1 Cost savings

Integration of applications and application serving products with WebSphere eXtreme Scale can help you scale your solution with minimal cost. Specifically, major cost savings can be realized in session offloading and dynamic cache offloading:

- ▶ Session offloading

Session offloading enhances efficiency in application performance and minimizes the need for additional JVM capacity to hold the session data. A specific example of this can be seen in the integration of WebSphere eXtreme Scale and WebSphere Portal Server. Offloading sessions from session heavy customer portlets can reduce the need for storage in the JVMs, thus increasing the efficiency of each JVM and reducing the need for additional Portal JVMs (and additional Portal licences).

- ▶ Dynamic cache offloading

The dynamic cache service in WebSphere Application Server is used extensively by WebSphere Commerce Server for caching pages, catalogs and other application critical objects. Offloading the dynamic cache from the JVM memory into eXtreme Scale grids reduces the load on the Commerce JVMs, reducing the need for additional JVMs and additional Commerce server licenses.

1.6.2 Adapting to growing business needs

Adapting your application infrastructure according to the future needs of your business is critical and requires precise engineering. Designing your scalability solution to be dynamic enough to keep pace with your growing business requirements is important for keeping your business competitive. WebSphere eXtreme Scale provides you with the capability to do this. Two major areas where eXtreme Scale can provide benefits for meeting business growth requirements are in extreme transaction processing and complex event processing

Extreme transaction processing

Scalability combined with performance is something every business wants. Conducting transactions using eXtreme Scale caching is much faster than transactions that span across multiple backend systems. Although positioning eXtreme Scale for extreme transaction processing requires changes to your existing architecture, the cost of this change is minimal compared to the benefits that can be realized over the long run.

Complex event processing

WebSphere eXtreme Scale can be positioned as the first layer in event processing architectures to filter out events which are not business relevant. eXtreme Scale also provide event enrichment, by efficiently caching event related data. Using WebSphere eXtreme Scale increases the scalability of the event processing platform by leveraging a routing optimization technique. The integration of WebSphere eXtreme Scale and WebSphere Business Events is a good example of this use.



Integrating WebSphere eXtreme Scale with other middleware

This chapter provides an overview of eXtreme Scale integration with existing IBM products. Business applications are not the only way to take advantage of the features of WebSphere eXtreme Scale. Several IBM products are also capable of taking advantage of these performance and scalability enhancements:

- ▶ WebSphere Commerce
- ▶ WebSphere Portal Server
- ▶ WebSphere Application Server
- ▶ WebSphere Business Events
- ▶ Rational Jazz

The plug and play features presently supported by eXtreme Scale on a integrated environment include:

- ▶ Scaling out the dynamic cache to a elastic, distributed and partitioned grid.
- ▶ Offloading session data from application servers to external grid containers.

This chapter includes the following sections:

- ▶ 2.1, “WebSphere Business Events and WebSphere eXtreme Scale” on page 16
- ▶ 2.2, “Rational Jazz based products and WebSphere eXtreme Scale” on page 21
- ▶ 2.3, “WebSphere Commerce and WebSphere eXtreme Scale” on page 23
- ▶ 2.4, “WebSphere Portal Server and WebSphere eXtreme Scale” on page 27

2.1 WebSphere Business Events and WebSphere eXtreme Scale

WebSphere Business Events provides an easy way to orchestrate and manage business events and real-time integration between disparate business systems. It is an event-driven application that can be configured to notify other systems to take appropriate actions when patterns of business events are detected.

- ▶ An *event* is an occurrence within a particular system or domain.
- ▶ An *action* is produced by WebSphere Business Events based on the evaluation of sets of business logic, in response to the events received.

Figure 2-1 shows a conceptual diagram of WebSphere Business Events, where events and actions flow between applications and the WebSphere Business Events process.

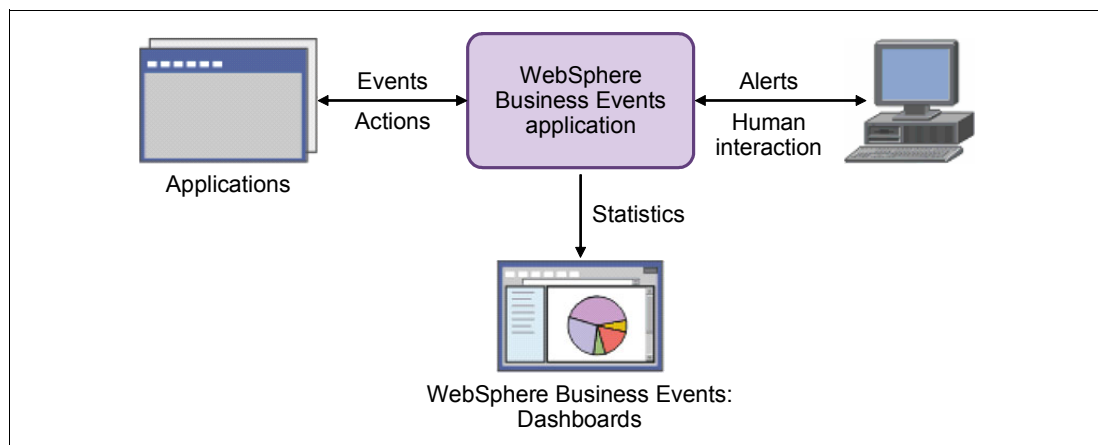


Figure 2-1 WebSphere Business Events architecture

WebSphere Business Events provides a dashboard that allows the real-time monitoring of business events and supports the ability to dynamically respond to trends based on threshold settings. The WebSphere Business Events User Console helps monitor human interaction in event flows.

2.1.1 Challenges

Handling large volumes of events and maintaining consistent throughput is a challenge that every event-driven application, including WebSphere Business Events, faces. Additional challenges exist when the application is operating in a distributed environment.

Handling large volumes of events

WebSphere Business Events can process and correlate many event types and can simplify complex event processing. In certain high volume event processing scenarios, many of the arriving events (or raw events) do not require the advanced event processing capabilities of WebSphere Business Events or are not even relevant for processing. Figure 2-2 on page 17 shows a high volume of raw business events thrown at the WebSphere Business Events process.

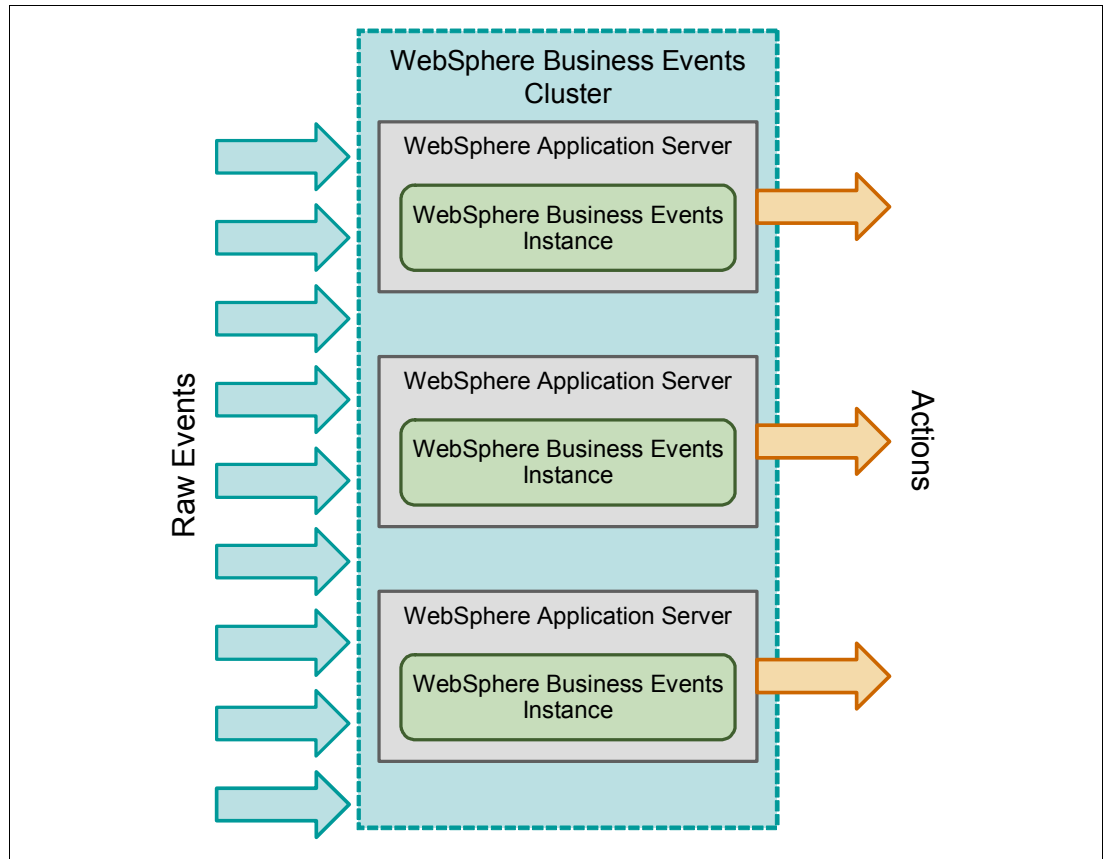


Figure 2-2 Large events hitting WebSphere Business Events cluster

In this scenario it might be efficient to perform pre-filtering of the raw events and only forward events that require more complex evaluation to WebSphere Business Events.

Event processing in a distributed environment

Complex event processing is the mechanism WebSphere Business Events uses to correlate business events for evaluation based on a pattern of events. A complex event is evaluated within a context, which is a set of interactions related by a unique identifier known as a context ID.

- An *interaction set* is the business logic that defines when an action is produced.
- *Touchpoints* are the business systems that interact with WebSphere Business Events through events and actions.
- A *result event* is a set of objects that can be returned from a touchpoint by a technology connector as the result of an action being sent to that touchpoint.

Figure 2-3 on page 18 shows the runtime environment of a WebSphere Business Events cluster processing a complex event. WebSphere Business Events uses data from two primary sources. The data can come from the event itself or be provided through WebSphere Business Events enrichment capabilities such as retrieval from a external data source. The WebSphere Business Events dashboard helps monitor the runtime events through charts. There are also history charts that can help you get the history data for analysis. The WebSphere Business Events user console helps you monitor human interaction in the event flow. WebSphere Business Events also provides a graphical console for administration.

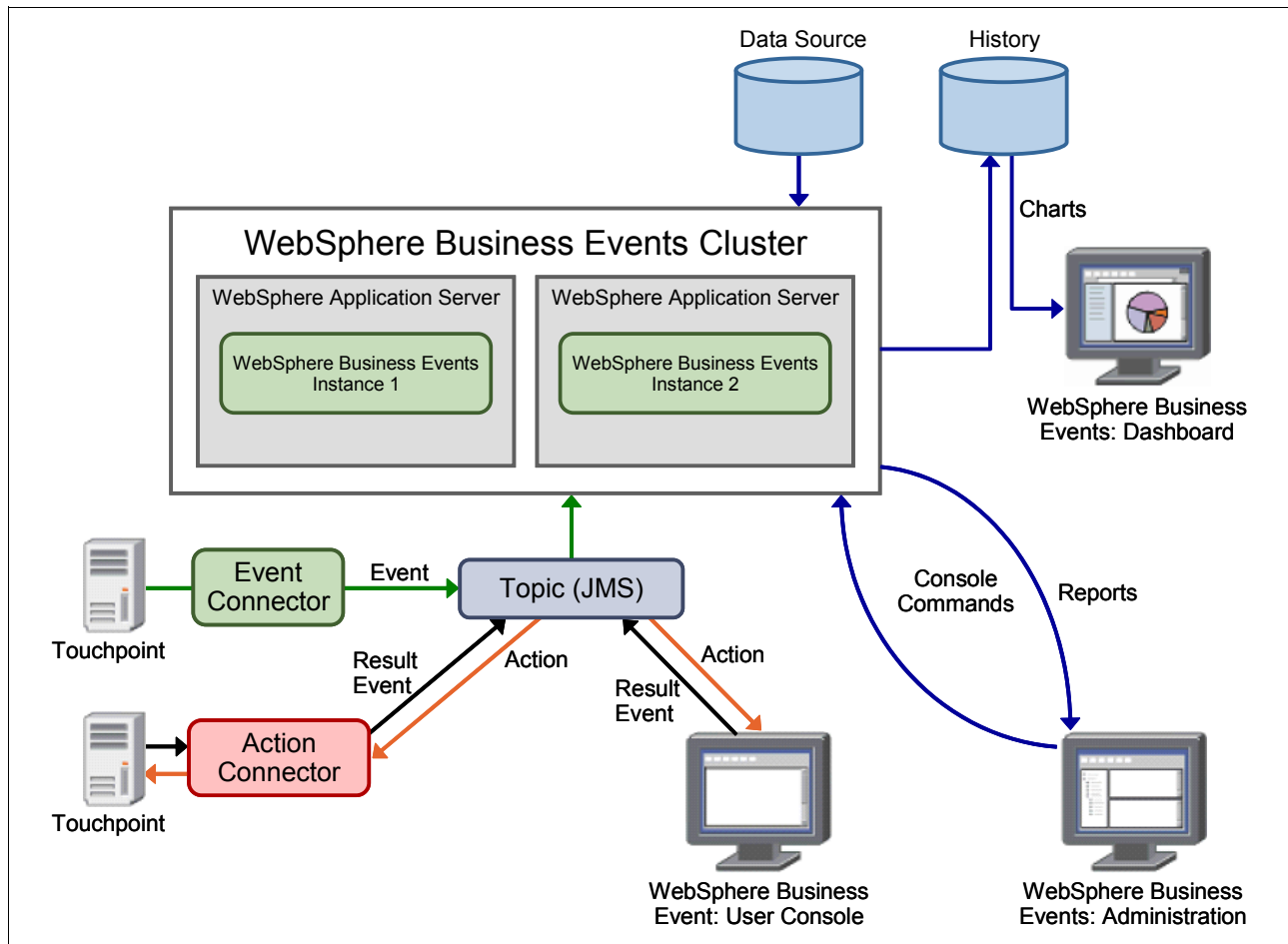


Figure 2-3 Complex event processing using a JMS topic and WebSphere Business Events cluster

When an event is received by WebSphere Business Events cluster, relevant data is passed by the touchpoint to the JMS queue or topic and can be delivered to any of the runtime members. To ensure proper functioning of complex event processing, events with the same context ID must be processed by the same WebSphere Business Events cluster member. To ensure this affinity, WebSphere Business Events transfers events between members if an event arrives on a server other than the one that is responsible for that context ID. This process requires serialization of events, which is expensive and can limit the scalability of the system.

2.1.2 WebSphere eXtreme Scale benefits

WebSphere eXtreme Scale with WebSphere Business Events (referred to as IBM WebSphere Business Event eXtreme Scale) can be used to address the challenges of high volume and affinity.

Combined offering: IBM WebSphere Business Events eXtreme Scale is a combined offering of WebSphere Business Events and WebSphere eXtreme Scale.

Using WebSphere eXtreme Scale as a pre-filter

WebSphere eXtreme Scale provides a distributed caching mechanism to store large amounts of data in memory and to access that data efficiently. It also exposes application programming

interfaces (APIs) to communicate with the cache from the Java code. Figure 2-4, shows how WebSphere eXtreme Scale can be positioned as the first layer in the business event processing architecture to filter the raw events. eXtreme Scale acts a damper, buffering WebSphere Business Events from a large percentage of the incoming raw events, many of which are not relevant to the event pattern detection processing of WebSphere Business Events.

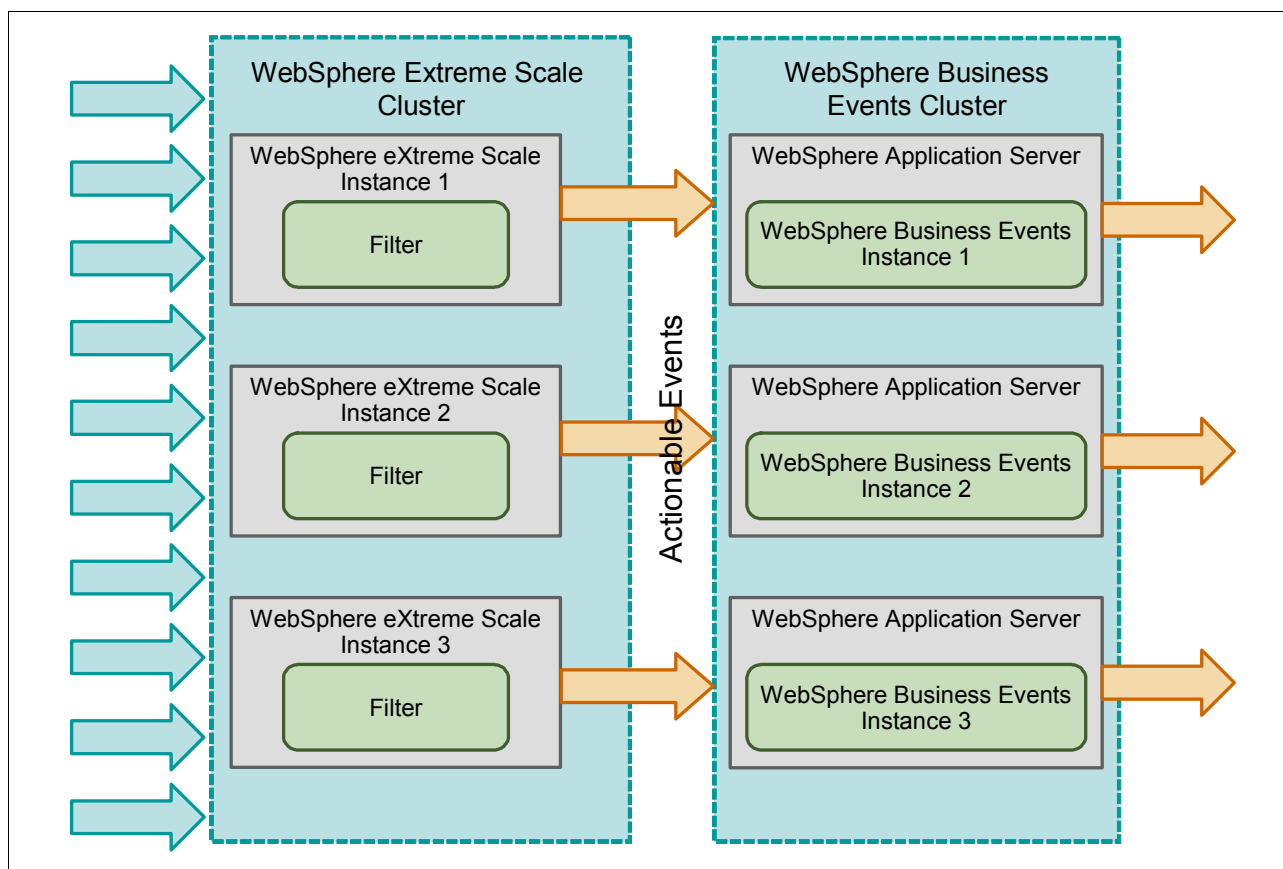


Figure 2-4 Using WebSphere eXtreme Scale for pre-Processing raw events

This solution requires you to segregate the existing event processing mechanism into two categories:

- ▶ Simple filtration logic

Simple checks are performed to decide if the event requires the complex event processing feature of WebSphere Business Events. This logic can be easily coded into the WebSphere eXtreme Scale grid. WebSphere eXtreme Scale can also provide event enrichment, by efficiently caching event related data.
- ▶ Complex event processing logic

Complex event processing is the job of WebSphere Business Events and should be created using the visual programming model available with WebSphere Business Events.

Using WebSphere eXtreme Scale for optimized routing

One of the ways through which WebSphere eXtreme Scale increases the scalability of the WebSphere event processing platform is by leveraging a routing optimization technique. This optimization technique bypasses the default JMS queue or topic used for event input and directly injects the event into the cluster member responsible for that event's associated contextID. This improves system efficiency by reducing the amount of internal routing that is

performed in the standard event flow path of WebSphere Business Events. Figure 2-5 depicts the usage of eXtreme Scale for optimized routing of business events.

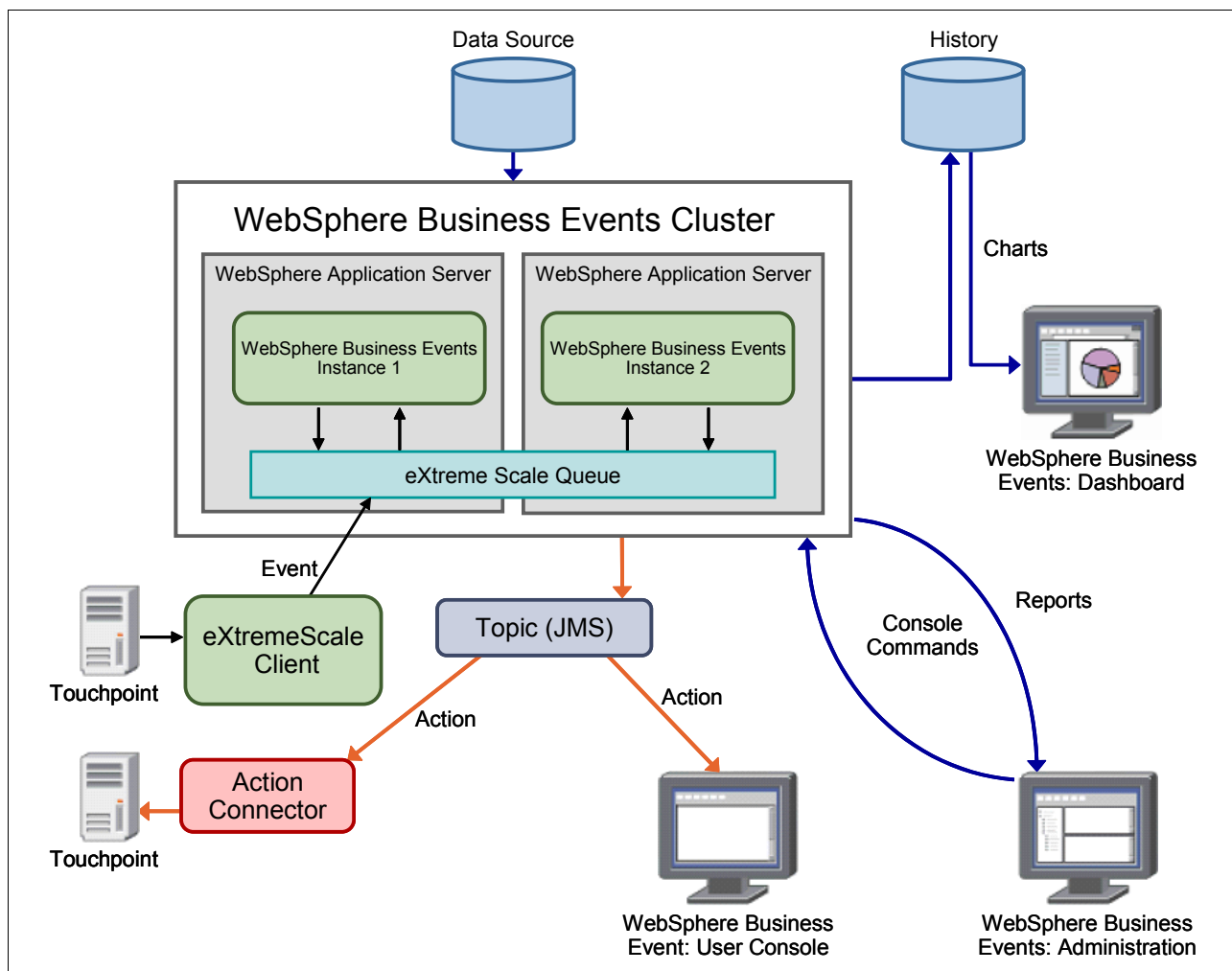


Figure 2-5 Complex event processing using eXtreme Scale and WebSphere Business Events

In this approach WebSphere eXtreme Scale provides APIs to send events directly to the correct WebSphere Business Events cluster member. The optimized routing mechanism requires that the contextID of the event with respect to the business logic can be determined prior to forwarding to WebSphere Business Events. The optimized routing mechanism reduces latency, but does not provide guaranteed message delivery.

Both the pre-filtering and optimized routing of the events can be used together in a single WebSphere Business Events environment.

2.1.3 For more information

Additional information about how WebSphere eXtreme Scale and WebSphere Business Events can work together can be found at Integrating WebSphere Business Events with WebSphere eXtreme Scale located at:

<http://publib.boulder.ibm.com/infocenter/wbevents/v7r0m1/topic/com.ibm.wbe.integrating.doc/doc/integrating-wxs.html>

2.2 Rational Jazz based products and WebSphere eXtreme Scale

This section provides an overview of the existing architecture, the challenges we face, and the benefits of eXtreme Scale integration with Rational Jazz based products.

2.2.1 Challenges

IBM Rational Jazz is a new technology platform for collaborative software delivery. Jazz is designed to transform how people work together to build software, making software delivery more collaborative, productive, and transparent. Product offerings that are built on the Jazz platform provide a rich set of capabilities for team-based software development and delivery.

Jazz-based products are designed for use as a single server configuration and cannot be used in either a cloned or a clustered configuration, except when implemented in an idle standby configuration, as shown in Figure 2-6. In this figure, you can see a Jazz-based product deployed on two nodes of a WebSphere Application Server cluster. At any given time, only one member is active and the others are in standby mode.

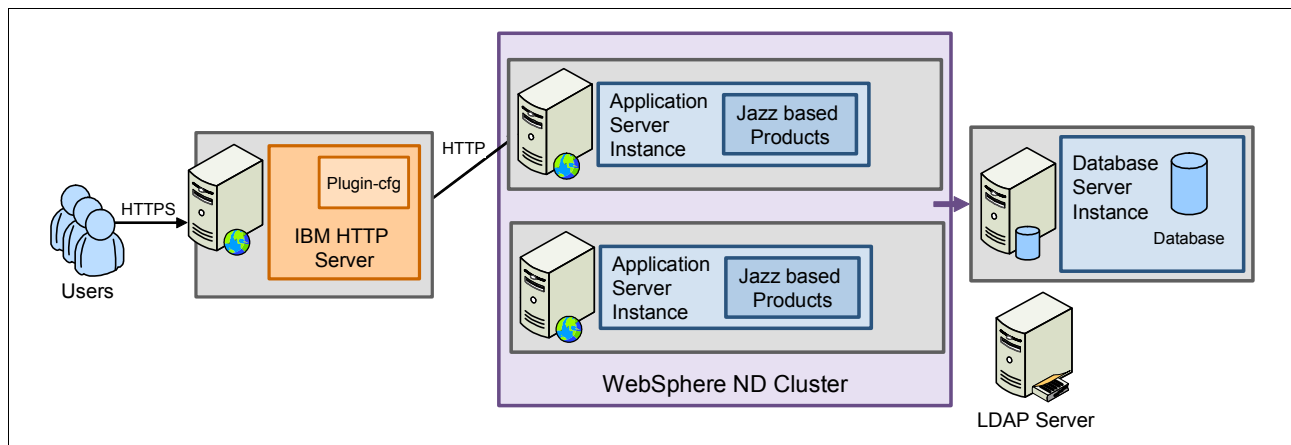


Figure 2-6 Existing cluster architecture for Rational Jazz with WebSphere Application Server

An idle standby configuration is not intended to provide complete support for failover. If the primary server fails or if it is intentionally taken offline, users will need to authenticate to the web application again or wait for their client to refresh a view.

Rational products that use the Jazz framework fetch the state data from a database and maintain that data inside their Java virtual machine (JVM). This state data is used to implement business requirements accordingly. If the primary server fails, the standby server needs to fetch all of the data from the database to create a new copy in its JVM. Also, the modified data in primary server cache can be lost.

2.2.2 WebSphere eXtreme Scale benefits

WebSphere eXtreme Scale can be used as a cache for state data, enabling Jazz based applications to run on clustered application server configurations.

Note: IBM Rational Team Concert™, Rational Quality Manager, Rational Requirements Composer, and Rational Asset Manager are the first offerings built on the Jazz platform. Although the Jazz Foundation platform has implemented the technology to take advantage of integration with WebSphere eXtreme Scale, these products have not been updated to use this technology. The integration strategy uses currently available products for IBM customer-written offerings.

Figure 2-7 shows how WebSphere eXtreme Scale can be integrated into a Jazz environment. WebSphere Application Server, along with the Jazz server code, provides the application servers to host the Jazz based products. These servers are clustered for load management and high availability. WebSphere eXtreme Scale is installed into the environment to provide the cache for state data.

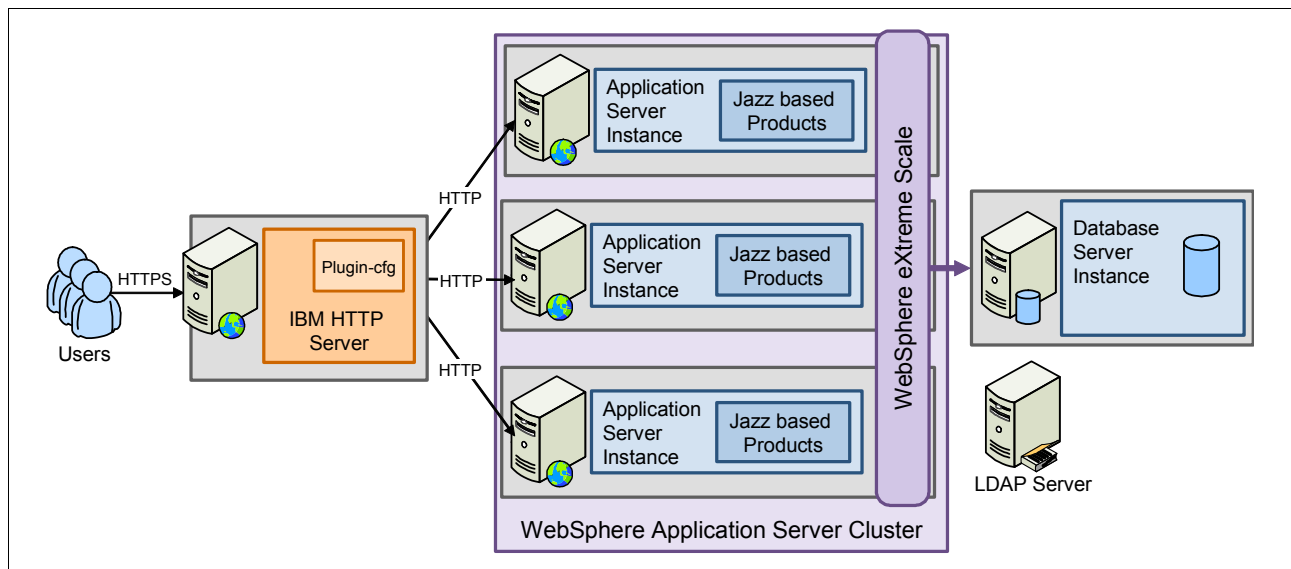


Figure 2-7 Clustering using WebSphere eXtreme Scale with Jazz based products

Using a clustered environment provides:

- High availability

By introducing WebSphere eXtreme Scale with a WebSphere Application Server cluster, all server nodes can work as active nodes. All of the nodes use the same state data managed in the eXtreme Scale JVMs. In the case of a failover situation, user requests are routed to a separate server using the HTTP server plug-in configuration. Session management can be achieved by enabling the single sign-on feature of WebSphere Application Server. This configuration helps system administrators to design better strategies for load balancing, load distribution, and failover.

- State data recovery

WebSphere eXtreme Scale can be used to provide shared memory for distributed state management to applications that are built on the Jazz platform. This allows these products to work on the same copy of the state data with fewer database operations. Shared memory also helps in reducing overall operation time.

2.2.3 Where to find more information

More information about deploying Rational Jazz applications in a clustered application server environment using WebSphere eXtreme Scale can be found at the Jazz community site. You will need to register at this site to see the article describing how to configure a Rational Jazz environment to use application server clustering with WebSphere eXtreme Scale. Registration is available at no cost. The article can be found at:

<https://jazz.net/wiki/bin/view/Main/DeployingJazzClusterWithWASND>

2.3 WebSphere Commerce and WebSphere eXtreme Scale

In this section, we provide an overview of the WebSphere Commerce caching architecture, the challenges faced in a WebSphere Commerce environment, and the benefits of eXtreme Scale integration with Commerce.

2.3.1 Challenges

The WebSphere Commerce Server is the server that handles the store- and commerce-related functions of an e-commerce solution. The WebSphere Commerce Server provides all of the WebSphere Commerce functionality in a web container and an EJB container. The WebSphere Commerce Server uses the WebSphere Application Server dynamic cache service to cache web and catalog content. Figure 2-8 on page 24 shows a typical deployment of the default dynamic cache provider. Each Commerce application server contains the same cached data, which is replicated between the application servers on a best-effort basis to provide consistency of application performance.

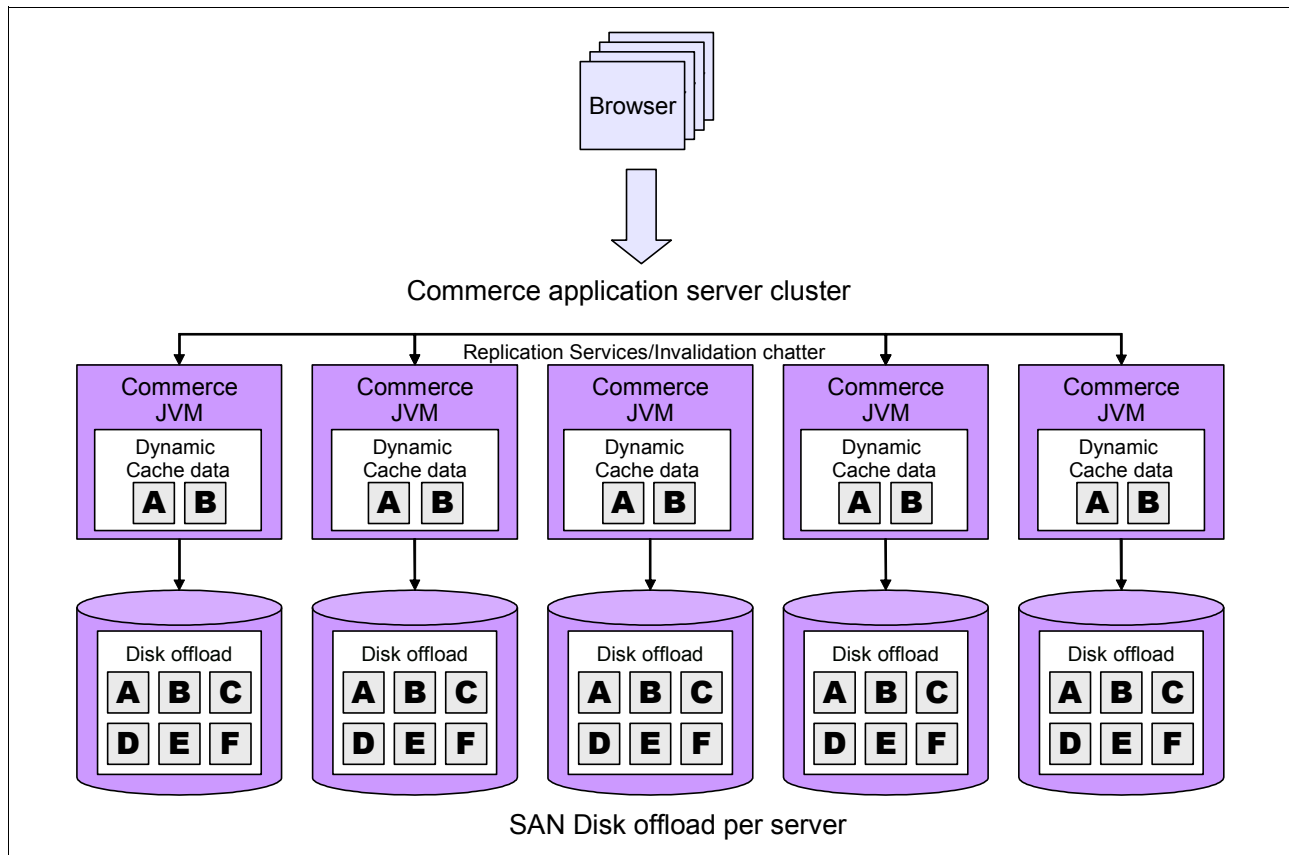


Figure 2-8 Traditional dynamic cache environment

The default dynamic cache architecture of the WebSphere Commerce Server poses the following challenges:

- **Scalability**

The dynamic cache capacity is determined by the individual JVM size. The scalability of the cache depends solely on how big an individual JVM can grow.

- **Performance bottleneck due to disk off-loading**

When the dynamic cache grows, it is practically impossible for a single JVM to hold all of the data, making it necessary to offload data to disk. Disk offloading decreases performance, and SAN or external storage is often required to offload the cache from the memory.

- **Commerce performance**

Commerce JVMs have large workloads to handle, in addition to holding cached data. A slight increase in the cache data exponentially affects the performance of the underlying JVM.

- **Cache duplication and high invalidation traffic**

The dynamic cache is a local cache, so to maintain consistency, the cache data is duplicated across all the cluster JVMs. This means that a change in dynamic cache content on one JVM creates invalidation traffic between the cluster JVMs.

- **Constraints on core group scalability**

Dynamic cache replication is done with the help of the data replication service (DRS) that runs in the underlying application server. The more members there are in a core group,

the greater the performance loss, due to the increase of DRS traffic between the cluster JVMs. Management of huge core groups becomes a challenge in itself. Additional management tasks are required to split the larger core groups into smaller ones using bridging to provide consistent data.

- Cold cache

With the dynamic cache, every restart of the JVM requires the data to be populated from the backend systems. This can consume a large amount of time. In the case of a lazy cache loading scenario, where only a subset of the cache is initially loaded, the optimal performance of using dynamic cache is only achieved after a significant amount of time.

2.3.2 WebSphere eXtreme Scale benefits

The primary benefit in using WebSphere eXtreme Scale is to reduce the total cost of ownership of running a WebSphere Commerce deployment. Figure 2-9 shows a typical deployment of the WebSphere eXtreme Scale dynamic cache provider.

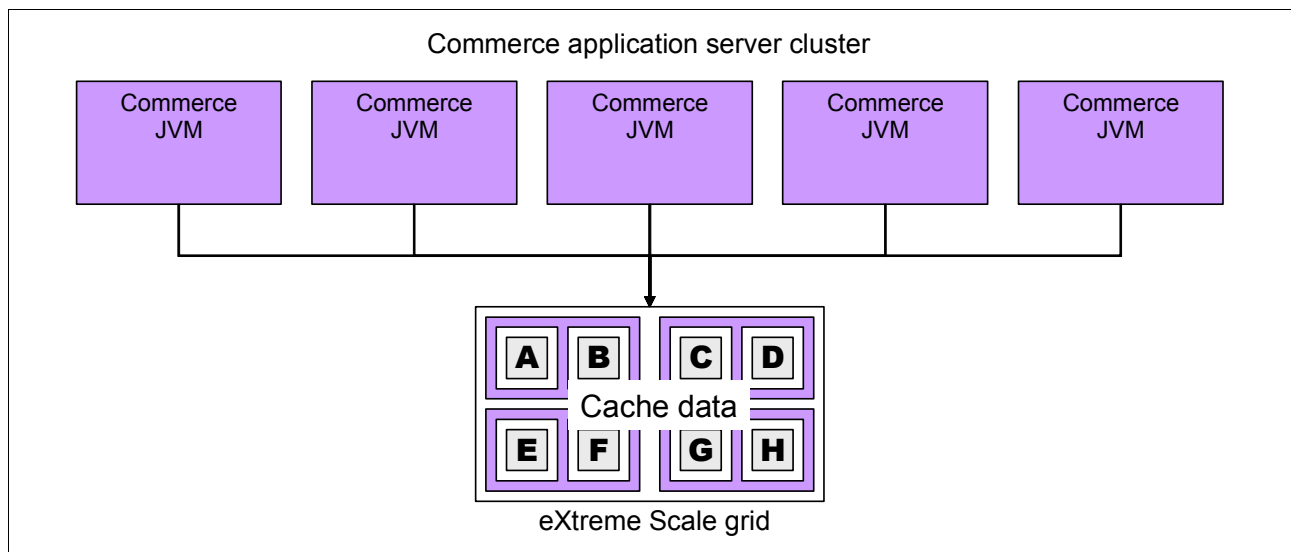


Figure 2-9 eXtreme Scale integrated into Commerce environment

By using WebSphere eXtreme Scale, you can make efficiency savings in a number of areas. The technical and business benefits include:

- Consistent and predictable performance as the site grows

The default dynamic cache provider in WebSphere Application Server is not designed to scale as effectively as eXtreme Scale. It is designed to copy data to other application servers, which naturally has its limits. WebSphere eXtreme Scale is an elastic, self-healing grid technology that has proven linear scalability of throughput and data quantity while retaining consistent high performance.

- A disk or SAN is not required

Eliminating this requirement provides cost, performance, and scalability benefits. A cache in eXtreme Scale can be very large, and an in-memory cache will be faster than a disk cache.

Consider how much disk space is needed. For example, the disk offload might be 20 Gb. This is duplicated n times for n servers. In our small example (Figure 2-9), this is 100 Gb of SAN requirement that is not needed for eXtreme Scale.

- WebSphere Commerce is more efficient

With eXtreme Scale, a Commerce application server can delegate all caching to another tier, and does not have to dedicate, for example, 30% of its memory to caching and a corresponding amount of CPU to manage cache replication, garbage collection and so on.

- Higher cache hit ratio and no duplication of cached data.

WebSphere eXtreme Scale stores the cached data in a single location, so all Commerce servers have access to the cache. Also, having no duplicated data removes the potential inconsistency of cached data where clients might receive different versions of cached pages from different servers. Having no duplication removes the communication needed to maintain cache consistency, making the cache more efficient in memory and CPU. And with no duplication of data there is less data to store. Compare this to dynamic cache. For example, each dynamic cache instance holds 400 Mb in memory, which is duplicated n times for n servers. For a small example of five Commerce servers (Figure 2-9 on page 25), this is 1.6 Gb of wasted memory.

- Consistency and predictability of site performance

One area where eXtreme Scale makes a big difference is when the Commerce servers need to be restarted. As part of the restart, the cache needs repopulating either by lazily recreating the cache, or by copying the data from another Commerce server. With either option there is a significant period after an application server restart where the site performance is well below optimal. WebSphere eXtreme Scale dramatically reduces site instability and warm-up time.

2.3.3 When to use WebSphere eXtreme Scale

The natural question is to ask, for what kind of environment should you consider using WebSphere eXtreme Scale? Unfortunately the answer depends on the environment, but there are clear criteria by which to decide. The following situations will derive benefit from using WebSphere eXtreme Scale:

- As soon as the cache is of a size such that no more than 50% can remain in server memory, you will benefit from greater and more consistent performance and less cache communication. Poor hit ratio can be determined by viewing the Cache Monitor.
- Large environments in particular benefit from guaranteed scaling and consistent performance. Large is subjective, but can apply from four application servers or more when you have exceeded the recommendation of four JVMs in a replication domain, or if experiencing a low cache hit ratio.
- Growing environments benefit from the elastic nature of eXtreme Scale, making it operationally straight forward to respond to the changeable nature of retail web usage, as a result of promotional activity or special occasions.
- If you cache large entries (such as large HTML pages), eXtreme Scale compression reduces the cache size.
- You need to ensure that no servers have stale data, so that end users do not see different cached pages (although the dynamic cache service provides 99% consistency).

Conversely, you probably will not need WebSphere eXtreme Scale if:

- You do not require access to the same cached data across a number of application servers.
- You do not require cache consistency across servers.

- You do not have much data to cache such that it easily fits into a single Java virtual machine including the application. In this scenario in particular, the default dynamic cache provider will likely be faster because all of the data is local.

2.3.4 Where to find more information

More information about integrating WebSphere Commerce with WebSphere eXtreme Scale can be found in Chapter 6, “Integrating WebSphere Commerce with WebSphere eXtreme Scale” on page 99.

2.4 WebSphere Portal Server and WebSphere eXtreme Scale

Traditionally, WebSphere Portal Server uses WebSphere Application Server’s infrastructure to ensure efficient session persistence management and configuration. However, this traditional approach has its limitations. In this section, we provide an overview of the existing WebSphere Portal Server architecture, challenges associated with it, and the benefits of WebSphere eXtreme Scale integration with custom-written portlets WebSphere Portal Server.

2.4.1 Challenges

One of the main design concerns of an application and application infrastructure is session persistence and handling of session state beyond the life of application servers or JVMs. Most enterprise Portal applications today require HTTP session persistence to provide high availability, session state handling, and application performance enhancements.

WebSphere Application Server’s underlying Data Replication Service (DRS) provides two mechanisms to persist sessions: memory-to-memory replication and shared database replication. Although these options exist to store HTTP session and state beyond the life of an application server or JVM, these approaches have operational challenges, performance and scalability concerns, and cost implications of their own. Although HTTP session replication enables seamless session recovery in the event of an application failure, it does not provide for a reliable, predictable, and cost effective solution to handle session recovery in the event of a data center failure. Use of session replication across cells and data centers has been discouraged primarily due to cost and performance implications.

Figure 2-10 on page 28 shows a typical Portal architecture that uses the DRS for memory-to-memory session replication.

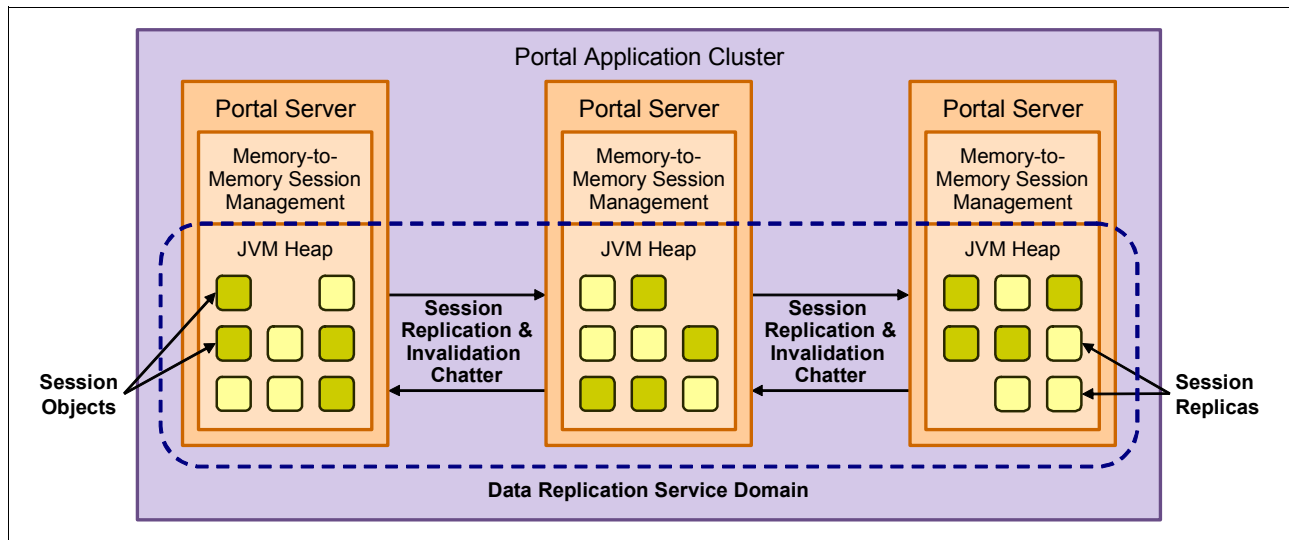


Figure 2-10 Memory-to-memory replication using WebSphere DRS

This architecture has the following challenges and limitations:

- ▶ Inefficient use of resources to store session data in the application server JVM heap
- ▶ Higher administrative and maintenance costs that come with an increasing user base and subsequent linear growth in hosting infrastructure
- ▶ Performance implications of storing and replicating session data due to serialization of the session object across application servers or JVMs (or even to a database), combined with cost of session state management like updates and invalidation
- ▶ Performance degradation that can occur in the case of large session objects

WebSphere eXtreme Scale addresses these challenges by providing a technological platform that supports improved management and replication as compared to the conventional HTTP session replication mechanism.

2.4.2 Integration architecture

WebSphere eXtreme Scale can be easily introduced into any WebSphere Portal Server architecture. Because it provides non-invasive integration for HTTP session management, the application does not need to be changed.

When eXtreme Scale is introduced into an existing Portal architecture, the WebSphere HTTP session manager is replaced by the eXtreme Scale session manager. An HTTP servlet filter, as shown in Figure 2-11 on page 29, intercepts every request prior to forwarding it to a servlet or JSP. The servlet filter wraps the `HttpServletRequest` and `HttpServletResponse` objects that the application developer uses to access the request's session state. The eXtreme Scale HTTP session object overrides the object normally provided by the default session manager. This way there is no data duplication between the two session managers and the eXtreme Scale session manager overrides the WebSphere session manager. The filter ensures that the session data is synchronized with the grid. If session attributes have changed, the request will be written back to the grid, ensuring consistency.

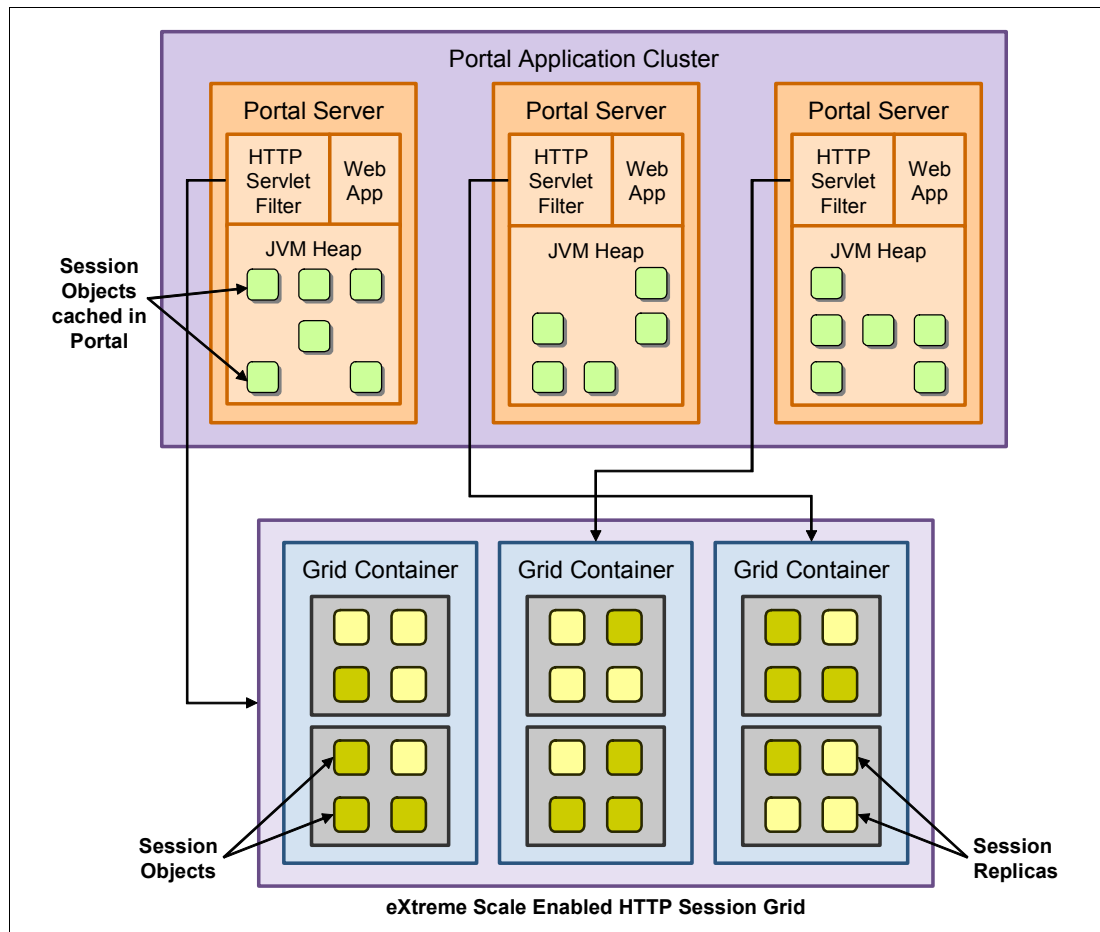


Figure 2-11 Session management using WebSphere eXtreme Scale

2.4.3 WebSphere eXtreme Scale benefits

Increasingly complex Portal application frameworks and changing dynamics of fast growing Portal applications demand a change in session persistence strategies and introduction of an in-memory data grid to provide a scalable and reliable solution to address the growth of session storage. WebSphere eXtreme Scale does exactly that. An in-memory grid provides an isolated layer whose sole purpose is to store HTTP session data. It helps overcome the challenges that exist with the traditional session replication strategy.

Because HTTP session memory-to-memory replication uses the underlying DRS of WebSphere Application Server, it is bound by the shortcomings of DRS. eXtreme Scale also employs a type of memory-to-memory replication of HTTP sessions, but is significantly better than DRS.

WebSphere eXtreme Scale provides the following benefits:

- ▶ Enables a grid of JVMs with a sole purpose of storing HTTP session objects.
- ▶ Isolates the application from the grid. It offloads session data to the grid, thereby freeing up the JVM for application use. By comparison, DRS is tied to the servers hosting the application.
- ▶ Provides linear scalability to accommodate growth in the number of sessions or size of session objects.

- ▶ Enables storing of session objects in replication domains in data centers spread across geographies and cells, unlike the single cell replication provided by DRS.
- ▶ Is easy to configure.
- ▶ Offers reliable replication and QoS as compared to the best effort protocol followed by DRS. Replication in WebSphere application Server does not guarantee consistency. If the system is overloaded, DRS will drop packets. eXtreme Scale ensures replication reliability and data resiliency. In DRS, after the replication domains are configured, they are static. In eXtreme Scale, even if a JVM is lost, the primary and replica placement happens dynamically.

2.4.4 Where to find more information

More information about using WebSphere eXtreme Scale for the management of session data used in custom-written portlets can be found in Chapter 5, “Integrating WebSphere Portal with WebSphere eXtreme Scale” on page 69.



WebSphere eXtreme Scale Architecture

This chapter provides an overview of the architectural structure of eXtreme Scale. Sections in this chapter include:

- ▶ 3.1, “WebSphere eXtreme Scale architecture” on page 32
- ▶ 3.2, “Grid client and servers” on page 35
- ▶ 3.3, “Catalog service” on page 36
- ▶ 3.4, “Shard placement” on page 37
- ▶ 3.5, “Zone support” on page 39
- ▶ 3.6, “Scalability sizing considerations” on page 40
- ▶ 3.7, “Common topologies” on page 41
- ▶ 3.8, “Monitoring eXtreme Scale” on page 47

3.1 WebSphere eXtreme Scale architecture

This section discusses the basic concepts required to understand how WebSphere eXtreme Scale is structured and how applications use it.

3.1.1 User view

The most fundamental component of the structure of WebSphere eXtreme Scale is the grid. Users connect to a grid, then access the maps in that grid. Data is stored as key value pairs in the maps.

Figure 3-1 shows a logical user view of a grid. Grid A has two maps, A and B, that store data.

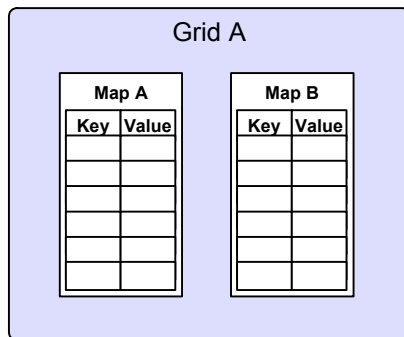


Figure 3-1 User view of a WebSphere eXtreme Scale grid

We use the term “grid” loosely. We have used it to refer to the entire WebSphere eXtreme Scale infrastructure, or that part that stores and manages data. However, the term grid has a precise technical definition in eXtreme Scale, namely, a container for maps that contain the grid’s data. Clients connect to grids, and access the data in the map sets they contain.

Figure 3-2 on page 33 exposes the next level of detail by introducing the map set and partitioning concepts. A map set is a collection of, or container for, maps. Therefore, a grid is really a collection of map sets. The key abstraction introduced by a map set is that it can be partitioned. This means it can be split into parts that can be spread over multiple containers.

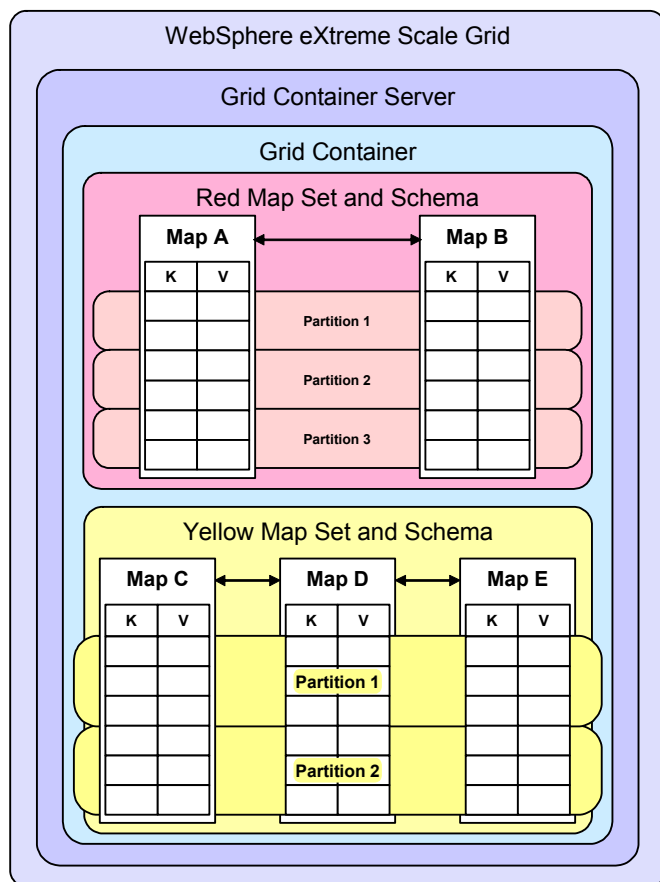


Figure 3-2 Maps in partitioned map sets in a grid

The following list provides detailed definitions for the components and terms used in WebSphere eXtreme Scale infrastructure.

- Map** An interface that stores data as key-value pairs (duplicate keys are not supported). A map is considered an associative data structure because it associates an object with a key. The key is the primary means of access to the data in the grid. The value part of the map is the data typically stored as a Java object.
- Key** A Java object instance that identifies a single cache value. Keys can never change and must implement the equals() and hashCode() methods.
- Value** A Java object instance that contains the cache data. The value is identified by the key and can be of any type.
- Map set** A collection of maps whose entries are logically related. More formally, a map set is a collection of maps that share a common partitioning scheme. Key elements of this scheme are the number of partitions, and the number of synchronous and asynchronous replicas.
- Grid** A collection of map sets.
- Partitions** The concept of splitting data into smaller sections. Partitioning allows the grid to store more data than can be accommodated in a single JVM. It is the fundamental concept that allows linear scaling. Partitioning happens at the map set level. The number of partitions is

specified when the map set is defined. Data in the maps is striped across the N partitions using the key hashcode modulo N . Choosing the number of partitions for your data is an important consideration when configuring and designing for a scalable infrastructure.

Shards

Partitions are logical concepts. The data in a partition is physically stored in shards. A partition always has a primary shard. If replicas are defined for a map set, each partition will also have that number of replica shards. Replicas provide availability for the data in the grid.

Grid containers

Grid containers host shards, and thus provide the physical storage for the grid data.

Grid container server

A grid container server hosts grid containers. It is WebSphere eXtreme Scale code running either in an application server or a stand-alone JSE JVM. Grid container servers are sometimes referred to as JVMs.

Figure 3-3 shows how these concepts are related.

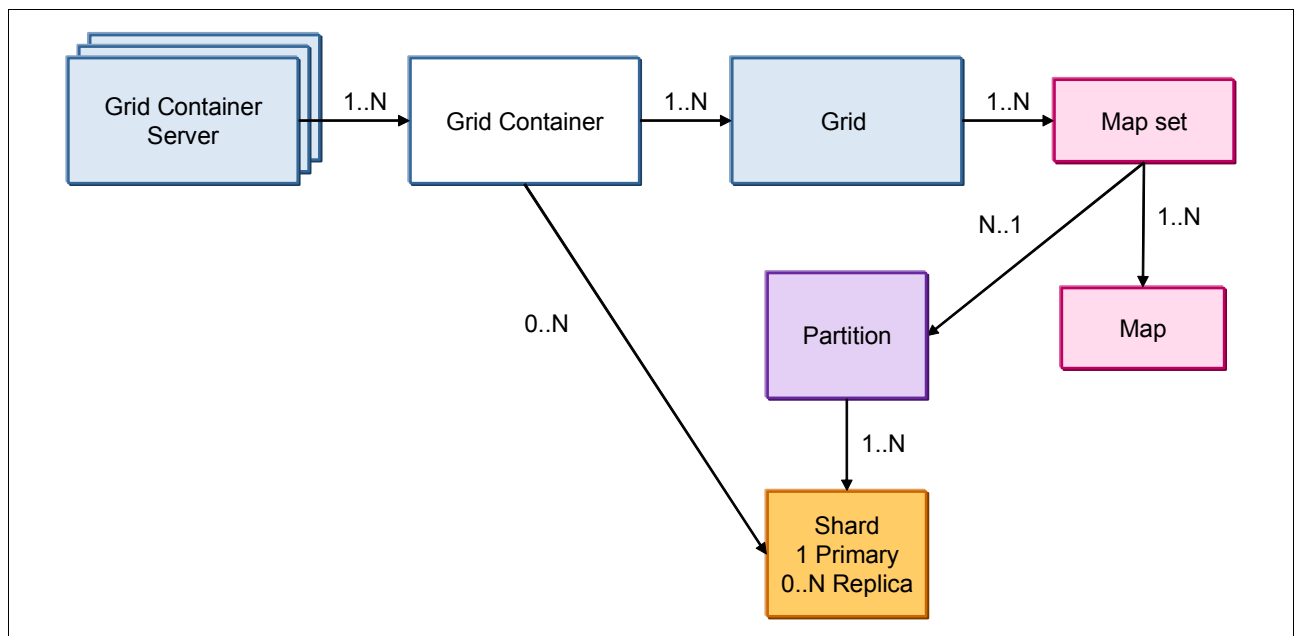


Figure 3-3 WebSphere eXtreme Scale meta mode

Note the following concepts:

- ▶ A grid container server can host many grid containers.
- ▶ A grid container hosts shards, which can be from one or more grids. A grid can be spread across many grid containers. The catalog service places shards in grid containers.
- ▶ A grid consists of a number of map sets. A map set is partitioned using a key. Each map in the map set is defined by a BackingMap.
- ▶ A map set is a collection of maps that are typically used together. Many map sets can exist in one grid.
- ▶ A map holds (grid) data as key value pairs.
- ▶ A map set consists of a number of partitions. Each partition has a primary shard and N replica shards.

Figure 3-4 shows how the parts and concepts we have discussed are interrelated. It shows how grid A can be physically stored in WebSphere eXtreme Scale. Grid A contains two map sets: Red and Yellow. The Red map set has three partitions (0, 1, and 2) and two replicas, for a total of nine shards (three partitions times three shards for each partition, the primary plus two replicas). The Yellow map set has four partitions (0, 1, 2, and 3) and one replica, for a total of eight shards. This is one way the shards can be distributed over the available grid containers.

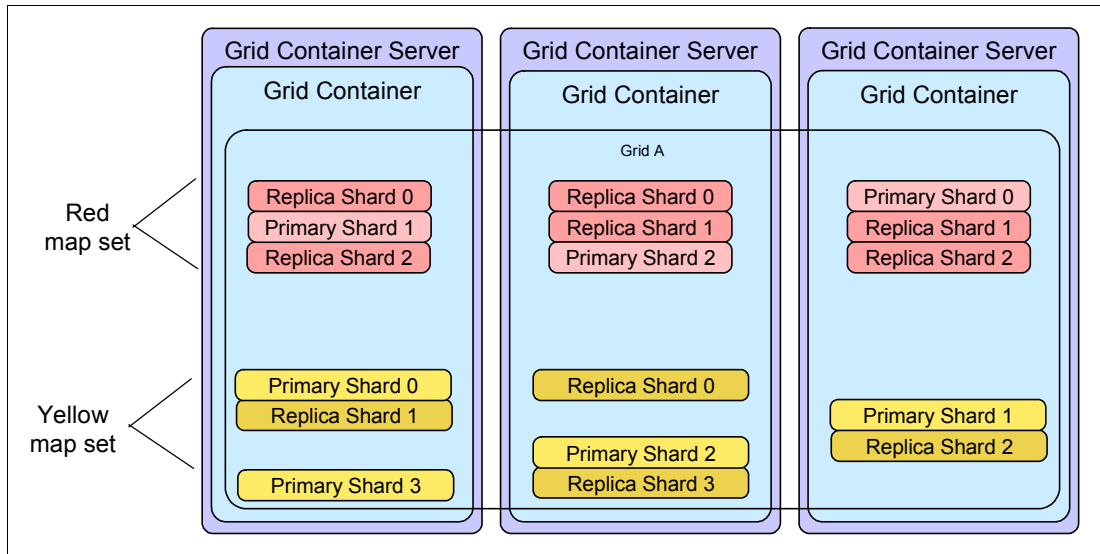


Figure 3-4 WebSphere eXtreme Scale grid

3.2 Grid client and servers

The following terms are used when discussing how an application interacts with the grid.

► Grid server

Catalog servers and the JVMs that host grid containers holding the cache are defined as grid servers. The catalog server's primary function is to provide routing information while the other grid servers host the cache (stored in BackingMaps).

Terms: The terms “grid server” and “ObjectGrid server” are interchangeable. In this, and other product documentation, particularly older documents, you will see ObjectGrid used at times.

► Grid client

Clients connect to a grid and are attached to the whole grid. Clients need to examine the key of application data to determine to which partition to route the request. Any entity that is attached to the grid with any kind of request becomes a client. A client contains an ObjectMap and can contain a near-cache copy of a BackingMap.

Terms: The terms “grid client” and “ObjectGrid client” are interchangeable.

A grid client can maintain an independent copy (near-cache) of a subset of the server data (far-cache). The far-cache is always shared between clients. The near-cache (if in use) is

shared between all threads of the grid client. Clients can read data from multiple partitions in a single transaction. However, clients can only update a single partition in a transaction.

Note: In our integration scenarios, we install the eXtreme Scale Client on WebSphere Portal Server and WebSphere Commerce Server so that it can communicate with a remote grid.

3.3 Catalog service

The catalog service, as shown in Figure 3-5, is a highly available service that maintains the healthy operation of grid servers and containers. The catalog service is made highly available by starting more than one catalog service process.

The catalog service becomes the central nervous system of the grid operation by providing the following essential operation services:

- ▶ Location service to all grid clients
- ▶ Health management of the catalog and grid container servers
- ▶ Shard distribution and re-distribution
- ▶ Policy and rule enforcement

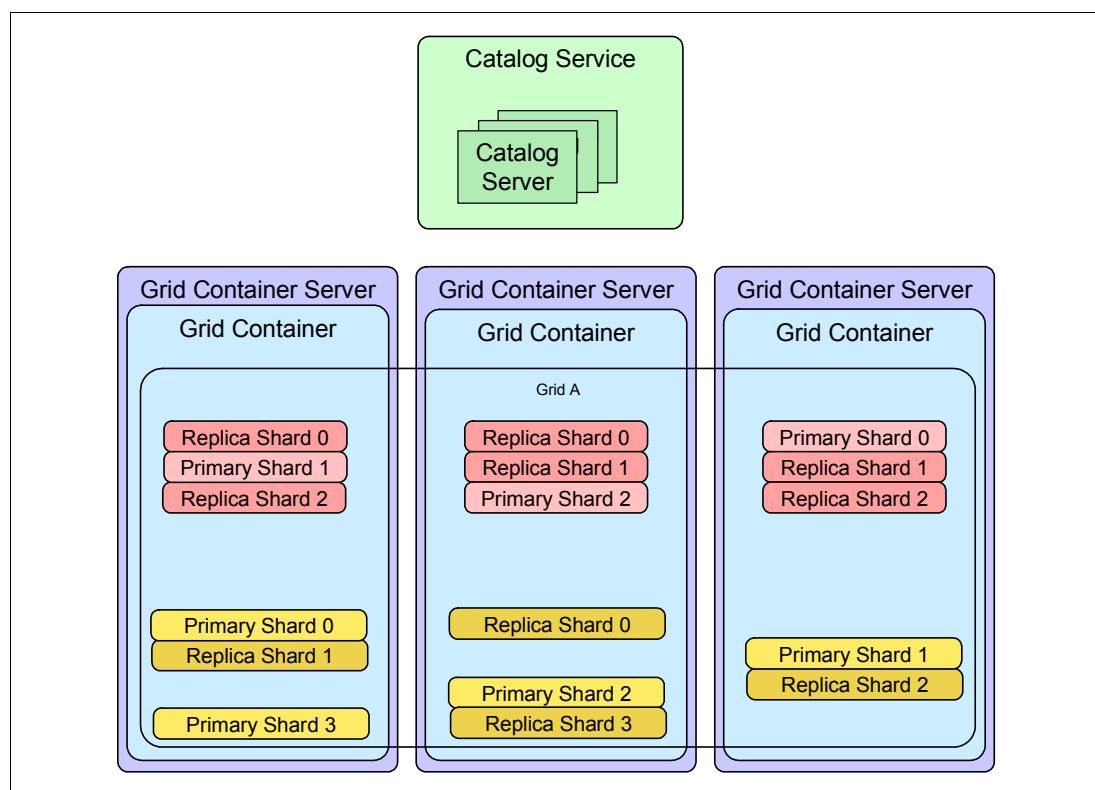


Figure 3-5 WebSphere eXtreme Scale grid

Because the catalog service's role is critical, it is necessary to consider the availability of the catalog service. The catalog service controls shard placement and routing for all clients, and should be clustered for high availability. This must be taken into account during the planning phases of the grid topology.

A catalog server can be configured and run in any JVM in the environment. A catalog server can be started as a stand-alone JVM or configured to be embedded into a WebSphere application server infrastructure process.

The catalog servers communicate together and typically rely on maintaining a quorum to determine the changes that need to be made to the grid configuration. Unless overridden, quorum is the full set of catalog servers. The catalog service will only respond to container events while the catalog service has quorum. Using this mechanism ensures that the grid configuration is not corrupted in the event of a catalog server loss due to a JVM or network failure. Intentionally stopping a catalog server instance does not cause loss of quorum.

When quorum is lost due to a temporary outage, it is re-established when the catalog server comes back online. When quorum is lost due to a more permanent or lengthy outage, the administrator can override quorum.

For more information, see *Catalog server quorums* at the following address:

<http://publib.boulder.ibm.com/infocenter/wxsinfoc/v7r0/topic/com.ibm.websphere.extremescale.over.doc/cxsquorcatsr.html>

Note: Designing high availability grids requires careful planning, sizing, and configuration. More details on high availability and failure modes can be found in the *Availability* topic of the WebSphere eXtreme Scale information center. This topic is available at the following web page:

<http://publib.boulder.ibm.com/infocenter/wxsinfoc/v7r0/topic/com.ibm.websphere.extremescale.over.doc/cxsavail.html>

3.4 Shard placement

The catalog service plays an instrumental role in the elastic nature of the grid configuration. It is responsible for replication, distribution, and assignment of the shards to grid containers. The catalog service evenly distributes the primary shards and their replicas among the registered container servers.

Water flow algorithm

The mechanism employed to distribute shards among the available grid containers is based on an algorithm resembling the natural flow of water. As grid container servers leave and join the grid, shards are re-distributed.

This re-distribution maintains shard placement rules, such as not placing primary and replica shards in the same container (or even the same machine, to maintain high availability). Section 3.5, “Zone support” on page 39 introduces another important shard placement rule.

It is important to understand the implications of the shard placement policy defined and enforced by the catalog service. The water flow algorithm ensures the equitable distribution of the total number of shards across the total number of available containers. Hence, WebSphere eXtreme Scale ensures that no one container is overloaded when other containers are available to host shards. eXtreme Scale also enables fault tolerance when the primary shard disappears (due to container failure or crash) by promoting a replica shard to primary shard. If a replica shard disappears, another is created and placed on an appropriate container, preserving the placement rules. Other key aspects of the approach are that it minimizes the number of shards moved and the time required to calculate the shard distribution changes. Both of these concerns are important to allowing linear scaling.

To ensure high (or continuous) availability of a data partition, WebSphere eXtreme Scale ensures that primary and replica shards of a partitions are never placed in the same container or even on the same machine.

Figure 3-6 shows four partitions, each with a primary shard and one replica shard, distributed across four containers.

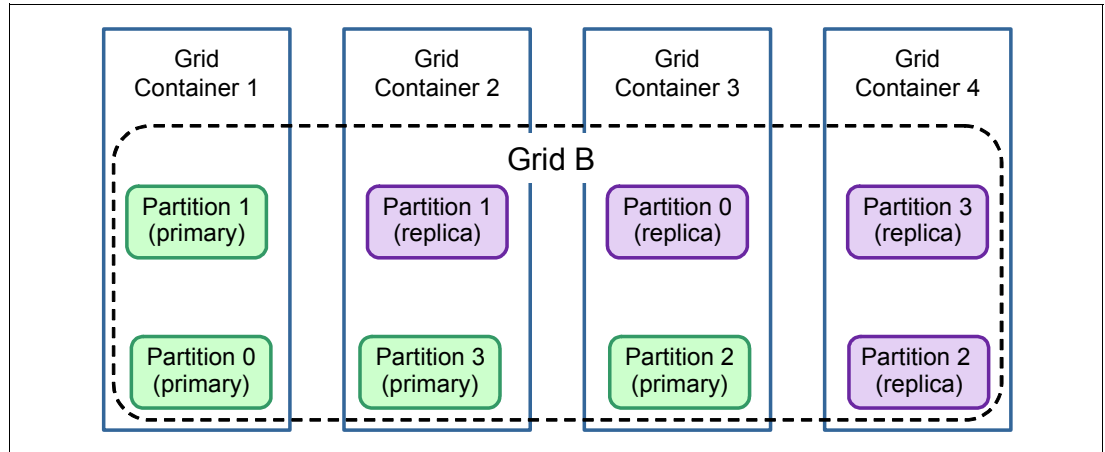


Figure 3-6 Shards placed on all available containers

Figure 3-7 shows how the shard placement adjusts when one of the containers fails and only three containers are available for placement. In this case, no primary partitions were affected so no replicas were promoted to be primary partitions. The two failed replica partitions are simply recreated in another container in the grid.

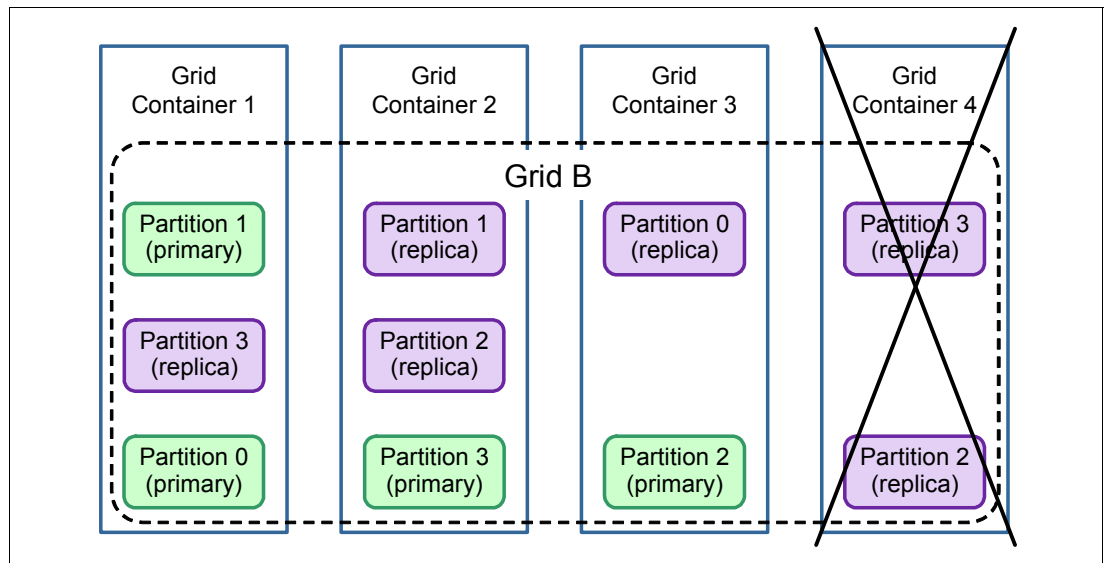


Figure 3-7 Shard placement and re-distribution after Grid Container 4 JVM Failure

3.5 Zone support

Zone support allows for rules-based shard placement, enabling high availability of the grid by placing replica shards across physical locations. This notion is particularly appealing to enterprise environments that need data replication and availability across geographically-dispersed data centers. In the past, these enterprise computing environments were limited due to the performance constraints imposed by networks and real-time data replication requirements.

With the inclusion of zone support, WebSphere eXtreme Scale offers better scalability by decoupling the grid environment. With zone support, only the metadata shared by the catalog servers is synchronously replicated. The data objects are copied asynchronously across networks. This not only enables better location awareness and access of objects, but also imposes fewer burdens on enterprise networks by eliminating the requirement of real time replication.

As long as the catalog service sees zones being registered (as the zoned grid container servers come alive), the primary and replica shards are striped across zones. Further, the zone rules described in the grid deployment descriptor file dictate placement of synchronous or asynchronous replica shards in respective zones.

As a general practice, place only synchronous replicas in the same zone and asynchronous replicas in a different zone for optimal replication performance. This placement is also optimal for scaling across geographies or data centers. This configuration also ensures high availability for a grid container server failure in a local zone (synchronous replica), and ensures high availability in a complete data center failure (asynchronous replica).

Typically, catalog servers are placed in each data center or zone, and the catalog servers synchronize their object/shard routing information. The catalog service must be clustered for high availability in every zone. The catalog service retains topology information of all of the containers and controls shard placement and routing for all clients.

This flexibility ensures the availability of data to the application regardless of its zoned location. The catalog service provides up-to-date routing information about the location of an object if the object is not found in the zone with the closest proximity to the application container.

Figure 3-8 on page 40 shows the relationship between zones and catalog servers.

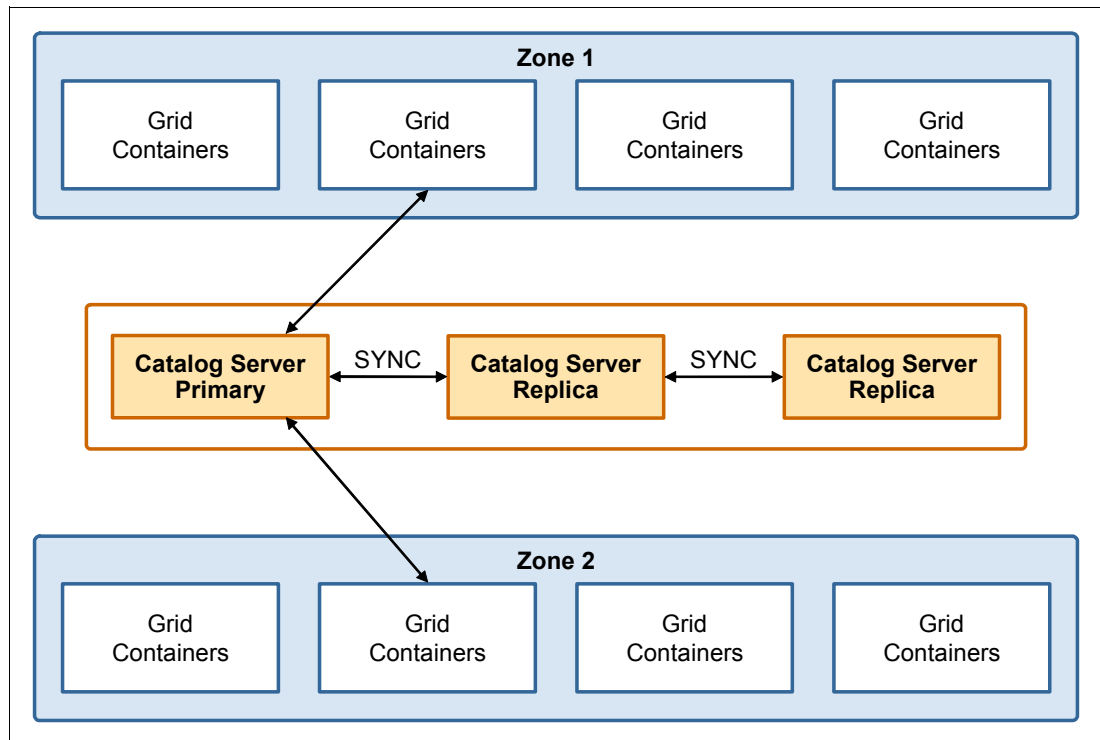


Figure 3-8 Zone placement

3.5.1 Zone-based routing

WebSphere eXtreme Scale provides a mechanism for clients to set preferences on how their requests are routed. eXtreme Scale supports a routing preference for zones, local host, and local process. This preference applies to both hash-based fixed partitions and per-container partitions.

Proximity-based routing provides the capacity to minimize client traffic across zone boundaries to minimize client traffic across machines, and to minimize client traffic across processes.

3.6 Scalability sizing considerations

WebSphere eXtreme Scale provides a scalable framework with a choice of runtime topologies to linearly extend the scalability of a data-driven transactional application. That is, the grid can grow elastically and transparently to grid operation as demand and data increase without increased overhead. This section provides high-level guidelines for sizing aimed at ensuring the scalability and reliability of a grid.

3.6.1 Heap size and the number of JVMs

This is probably the most common, and important consideration. Although the eXtreme Scale grid offers linear scalability, grid deployment, data size, and application patterns determine the size and volume of JVMs that make up the grid. This is also an important data point to facilitate future growth of the grid. Consider grid deployment factors such as estimated partition size, number of partitions, number of synchronous and asynchronous replicas,

desired availability, zoned deployment over geographies, and so forth, when deciding the total JVMs and size (in terms of memory allocation) of JVMs that make up the grid.

3.6.2 Number of grids

A single grid infrastructure can support many logical grid instances, each containing a distinct set of maps and data. A single application can connect to multiple grid instances if required. In general, assign each application its own dedicated grid instance, except in cases where data is shared between applications.

Put a different way, multiple applications and multiple grids can share the same catalog service. In general, provide each logical set of grids used by one or more applications its own dedicated catalog service to maintain a separation of applications so that one application's problems do not cause failures in the other applications.

3.6.3 Catalog servers

The catalog service plays a central role in eXtreme Scale grid management. It is therefore vital to plan for catalog service high availability and sizing requirements. The decision on the number of catalog servers in the catalog service depends on overall grid size, desired high availability, and zones configuration. As a general rule, use at least two catalog servers for high availability, and at least one per zone.

3.6.4 Sizing for growth

The appeal of WebSphere eXtreme Scale is the ability to scale linearly with growth. Therefore, factor in the growth imperatives for ease of grid infrastructure administration and for accommodation of growth. Growth imperatives include decisions and considerations regarding grid topology, hardware requirements and availability, and managed or stand-alone grid environment. The driving factors are the resource (hardware and software) availability and the set of tasks involved in adding grid containers to expand the grid on demand. This imperative can also include the decision points and operational procedures required to add to the grid capacity.

It is particularly important to set the number of partitions high enough to allow for future growth. After the application is deployed, the number of partitions cannot be changed without restarting the entire grid. This means that beyond the point where there is one only one shard on each grid container JVM, adding more JVMs will not provide any benefit. A good general rule is 10 shards per grid container JVM.

3.7 Common topologies

Before designing your topology, it is important to consider what type of software you will install on your servers to house your grid. There are two types of servers, stand-alone and managed by a deployment manager. More details on this can be found in Chapter 4, "Installing WebSphere eXtreme Scale" on page 57.

3.7.1 Offloading data from server JVMs

Offloading data to an external JVM is the preferred topology when you face memory constraints or huge memory requirements with the existing environment. As we mentioned

previously, WebSphere eXtreme Scale provides a highly scalable and elastic environment for large memory requirements and helps the application JVM work efficiently. In integration scenarios, you can offload dynamic caches and sessions from application JVMs to remote eXtreme Scale JVMs.

In this section, we compare two topologies: *remote managed eXtremeScale* and *remote stand-alone eXtremeScale*.

Remote managed eXtreme Scale topology

Figure 3-9 shows the grid installed in the WebSphere Application Server JVM and is remote to the application. Apart from the offloading feature, it also provides more flexibility and ease of management. We use this topology in our examples.

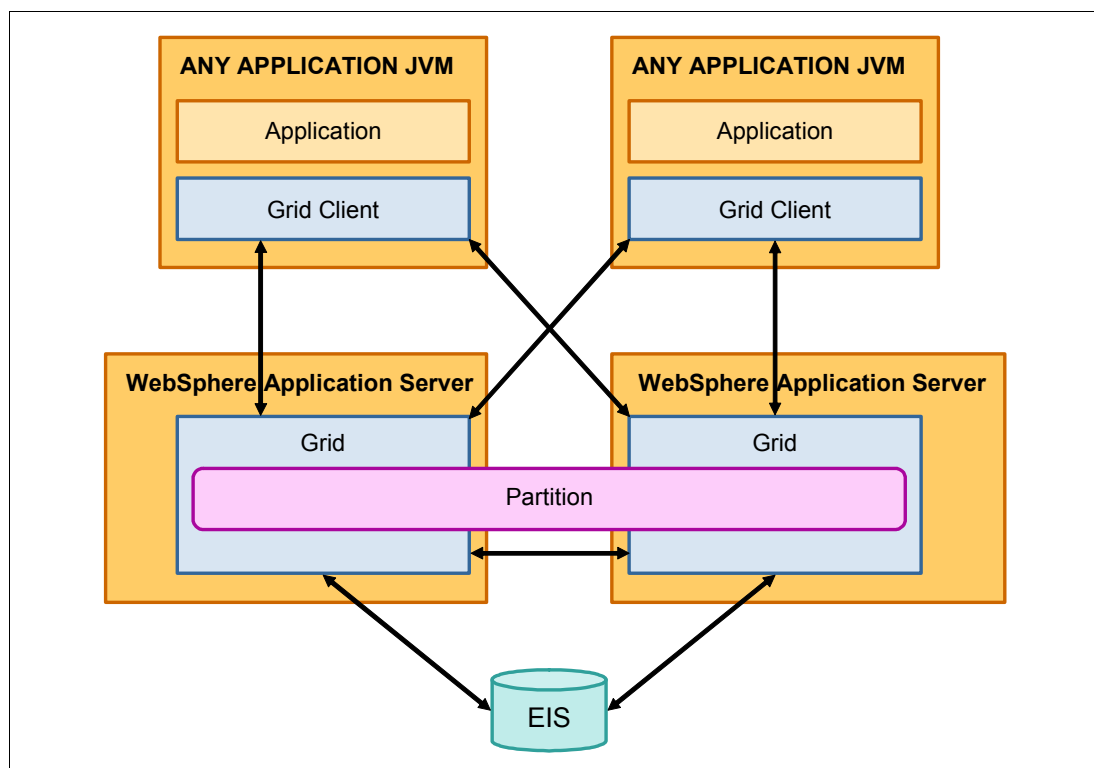


Figure 3-9 Remote and managed grid on a remote WebSphere Application Server JVM

Remote managed topology gives you the following advantages:

- ▶ Offloading of session or cache data from the application JVMs helps the application respond faster
- ▶ Better management facilities using the Network Deployment administrative console
- ▶ Clustering and high availability management is taken care of by WebSphere Network Deployment
- ▶ Adding and removing grid members is easier
- ▶ Zones can be easily set up using the nodegroup facility
- ▶ Existing monitoring tools of the WebSphere Network Deployment environment can be easily leveraged

A disadvantage when using remote managed topology is the cost factor. There are additional WebSphere Application Server license issues involved.

Remote stand-alone eXtreme Scale topology

Figure 3-10 shows a remote stand-alone eXtreme Scale topology. In this example, the grid client communicates to the remote stand-alone grid server. The WebSphere Application Server container does not host the grid. This setup provides all of the facilities provided by the embedded version except for the WebSphere Application Server monitoring features.

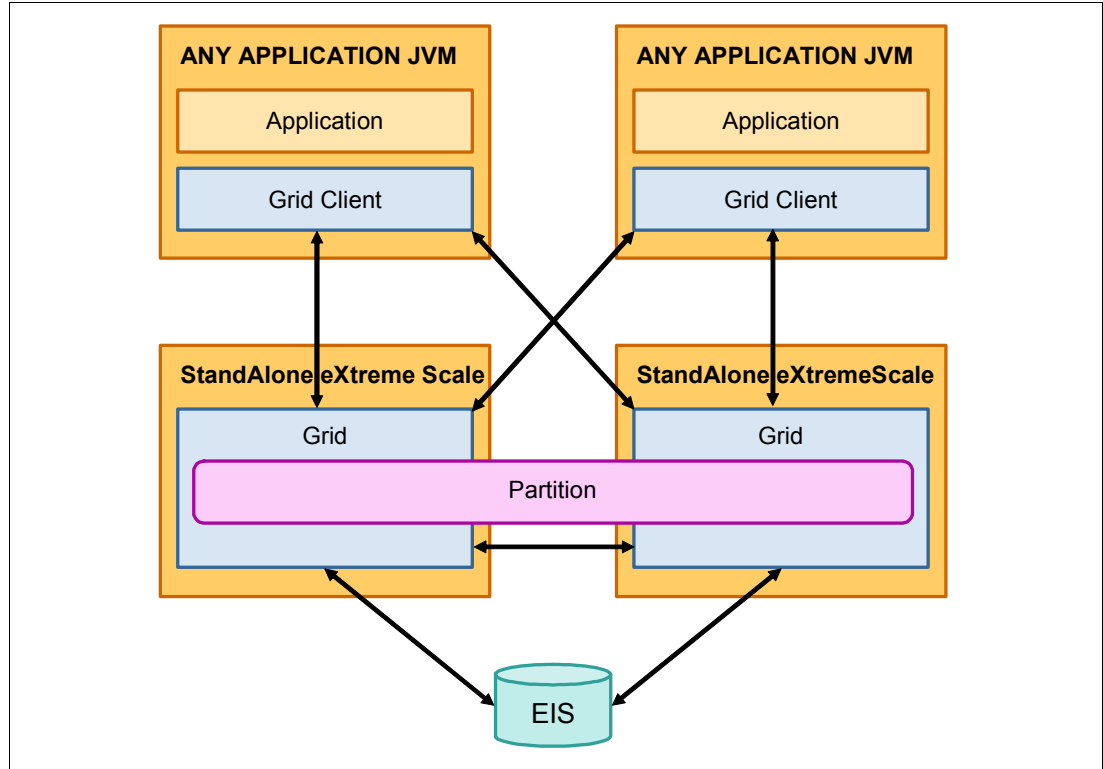


Figure 3-10 Remote stand-alone eXtreme Scale topology

Advantages when using remote stand-alone topology include:

- Cost savings because there are no WebSphere Application Server license issues involved.
- Session and dynamic cache offloading.

Disadvantages when using remote stand-alone topology include:

- Less administrative control due to the absence of WebSphere Application Server administrative console.
- Certain monitoring techniques are not supported in a stand-alone environment. Tivoli Performance Viewer and Cache Monitor runs only in a WebSphere Application Server environment.

3.7.2 Collocating cache with server JVMs

In this section, we compare two cache topologies: *embedded* and *embedded partitioned*.

Embedded topology

In the local cache topology shown in Figure 3-11 on page 44, the application logic runs in the same JVM as the data in the grid. Each application can only access the local grid instance to

store or retrieve data from its cache. In this case, WebSphere eXtreme Scale is used as a simple local cache.

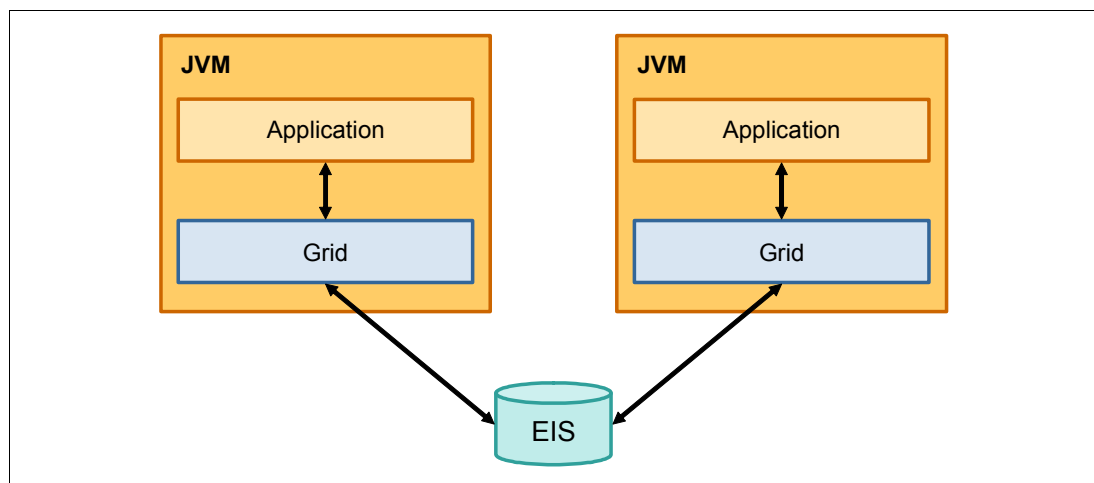


Figure 3-11 Local cache topology

This topology can perform faster than using a database if the needed data can be found in the local ObjectGrid. This avoids a remote procedure call (RPC) to the back-end data store. Using WebSphere eXtreme Scale as a local cache can also reduce the load on your back-end data store. This topology is not recommended for fault tolerance or high availability.

A big drawback with this approach is you have more than one cache for the data. In this case, you have to keep the caches in sync. If an object is cached in both JVMs and is changed in one of them, the other cache becomes outdated. The two caches must be set up to shared cache to resolve the object, which is difficult to set up. This topology also scales poorly as you add applications with local caches. Use the following topologies instead to use a shared cache and avoid the problem altogether. This topology works well only when the applications are reading the data and not making any changes.

Embedded partitioned topology

In the colocated application and cache topology shown in Figure 3-12 on page 45, the application logic runs in the same JVM as the data in the grid. However, the data stored in the grid is spread across all of the JVMs that have WebSphere eXtreme Scale installed and configured.

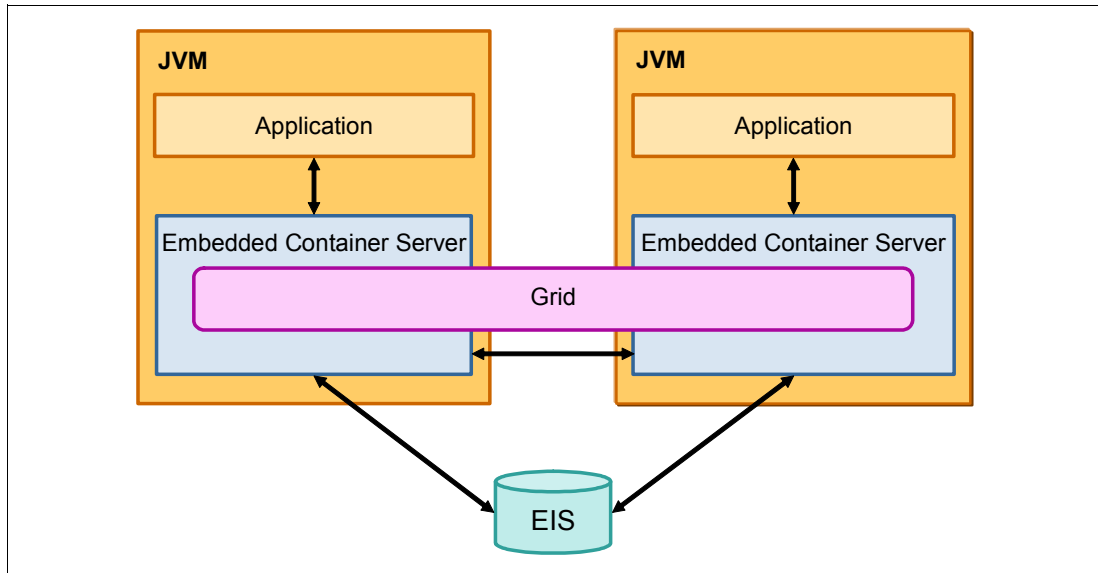


Figure 3-12 Co-located application and cache topology

This topology can be faster than using a database because an application can take advantage of the grid near-cache to compensate for the RPC calls made when the requested data is stored on another server in the grid or can only be found in the back-end datastore. This topology can also reduce the load on your back-end datastore.

With this topology, replica shards that sit in a JVM other than the primary can be used to ensure fault tolerance and high availability.

3.7.3 High availability and multiple data center topologies

A zone-based topology helps enterprise computing environments to distribute their data cache across geographies for high availability and disaster recovery motives. For the distributed application spanning across multiple data centers, a multi-master topology helps in reducing data access time. In this section, we compare two topologies: *zone-based* and *multi-master*.

Zone-based topology

Zone support provides much-needed control of shard placement in the WebSphere eXtreme Scale-enabled grid. By definition, zones can be considered as a set of grid containers that belong to a domain or exist within a boundary. Multiple zones can be envisioned to exist across WANs and LANs, or even in the same LAN, but one zone is not intended to span across a WAN. Instead, define multiple zones across a WAN and combine them to form one single grid. Such a topology includes the following advantages:

- ▶ High availability of data cache across geographies
- ▶ Proximity of data to the application
- ▶ Controlled rule-based replication
- ▶ Primary and replica can be placed in separate zones, satisfying the disaster recovery requirements

The replication of data in real time, such as HTTP session data and application data, was a concern in the past due to cost of network and computing resources. The effort and costs involved in achieving this type of replication outweighed the potential benefits. Slower network connections mean lower bandwidth and higher latency connections. Zone-based replication

factors include the possibility of network partitions, latencies, network congestion, and other factors.

A WebSphere eXtreme Scale grid adjusts to this unpredictable environment in the following ways:

- ▶ Limiting heartbeat to reduce traffic and processing
- ▶ Exploiting the catalog service as a centralized location service

Figure 3-13 shows a zone-based topology.

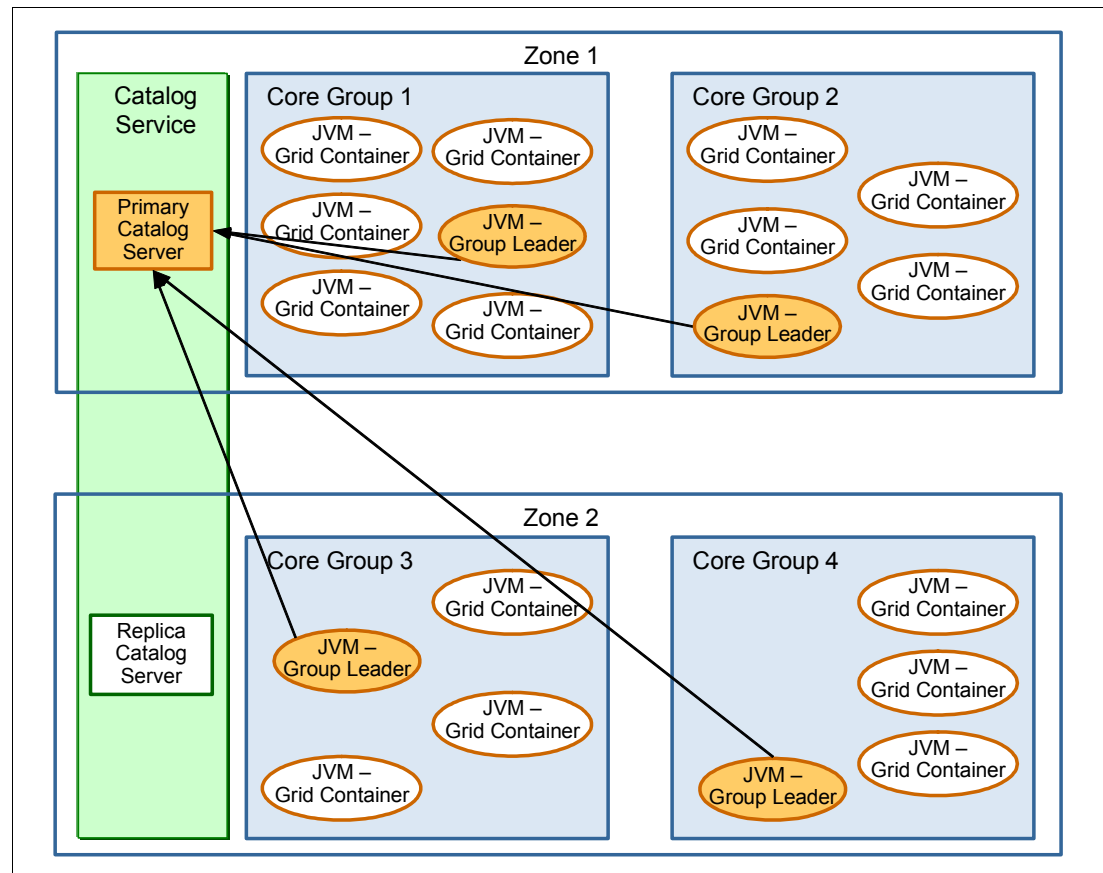


Figure 3-13 Zone-based topology

The catalog service uses the WebSphere Application Server high availability manager to group processes together for availability monitoring. Each grouping of the processes is a core group. With eXtreme Scale, the core group manager dynamically groups the processes together. These processes are kept small to allow for scalability. Each core group elects a leader that has the added responsibility of sending status to the core group manager when individual members fail. The same status mechanism is used to discover when all the members of a group fail, which causes the communication with the leader to fail.

Multi-master topology

Using the multi-master asynchronous replication feature, two or more data grids can become exact mirrors of one another. This mirroring is accomplished using asynchronous replication among links connecting the grids together. Each grid is hosted in an independent catalog service domain, with its own catalog service, container servers, and a unique name.

With the multi-master asynchronous replication feature, you can use links to interconnect a collection of these catalog service domains and then synchronize the catalog service domains, using replication over the links. Such a topology includes the following advantages:

- ▶ Availability and partition tolerance.
- ▶ eXtreme Scale uses an arbitrator to resolve collisions.
- ▶ Replication is not queue based. There is no concern if the link is down.
- ▶ Each container JVM communicates directly to the shards in the other location.
- ▶ Lower response time for user queries.

In a distributed environment, especially those that span data centers across a LAN and WAN, management of client response time is a challenge. As shown in Figure 3-14, response time for the client accessing data from a data center located in a separate geographic region can vary greatly as compared to accessing data from a data center located in its own geographic region. In a multi-master topology, every domain can have its own primary replica. Every domain asynchronously replicates data with its linked domains. Data written to one primary is asynchronously replicated to linked primaries as well. This data can be read locally from any data center.

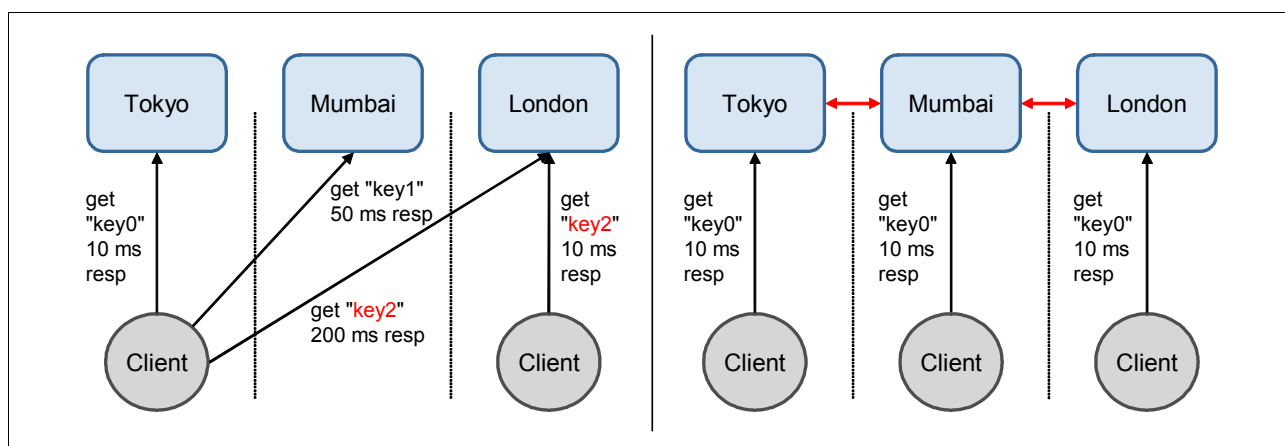


Figure 3-14 Multi-master topology

3.8 Monitoring eXtreme Scale

In this section, we discuss the techniques and utilities used to monitor WebSphere eXtreme Scale. Monitoring of session data for WebSphere Portal Server is discussed in more detail in 5.4, “Monitoring” on page 92. Monitoring the dynamic cache in WebSphere Commerce Server environments is discussed in more detail in 6.4, “Monitoring considerations” on page 132.

3.8.1 Using the eXtreme Scale web console

The WebSphere eXtreme Scale web console is a new feature with version 7.1. It is primarily used for sizing and performance related information. The web console user interface supports the following web browsers:

- ▶ Mozilla Firefox 3.6.x and later
- ▶ Microsoft® Internet Explorer 7 and later

The WebSphere eXtreme Scale web console comes with the stand-alone installation of WebSphere eXtreme Scale. Because the console is new in version 7.1, we will briefly cover the steps to get it up and running:

1. Start the web console by running the following command:

```
WXS install\ObjectGrid\bin\startConsoleServer.bat
```

2. Enter the console URL in a web browser:

```
http://localhost:7080
```

3. Log in with the user name and password `admin`. This password can be changed later from the console after the first login.

4. Configure the web console to use your catalog servers. In the console, click **Settings** → **eXtreme Scale Catalog Servers** and add the catalog server:

- Host Name: `think`
- JMX Port: `2809`
- Listener Port: `2809`

There is a slight difference in configuring a stand-alone catalog server and an embedded catalog server on the console.

- Standalone catalog server

For stand-alone servers, the default JMX port for the catalog server is 1099. These settings can be changed by passing the parameters `-JMXServicePort` and `-listenerPort` during the startup of the catalog server. See Section 4.2.2, “Starting a catalog server on a stand-alone installation” on page 66 for more information about starting the catalog server.

- Embedded catalog server

A catalog server can run in a WebSphere Application Server process, and is typically placed in a Deployment Manager or Node Agent. For a catalog server hosted in WebSphere Application Server, the JMX Port and Listener Port are the bootstrap port of the catalog server process. (The Deployment Manager defaults to 9809 and the Node Agents default to 2809).

5. In the console, click **Settings** → **eXtreme Scale Domains** and create a new catalog service domain. Provide an arbitrary name and include the catalog server you created previously.

Tip: If the console doesn’t connect to a catalog server, check the connectivity on the port. If it continues to fail to connect, restart the console from the command line using the following command:

```
WXS install\ObjectGrid\bin\startConsoleServer.bat
```

6. View the reports on the monitoring tab. Figure 3-15 on page 49 shows a Sample Grid Usage View from the web console.

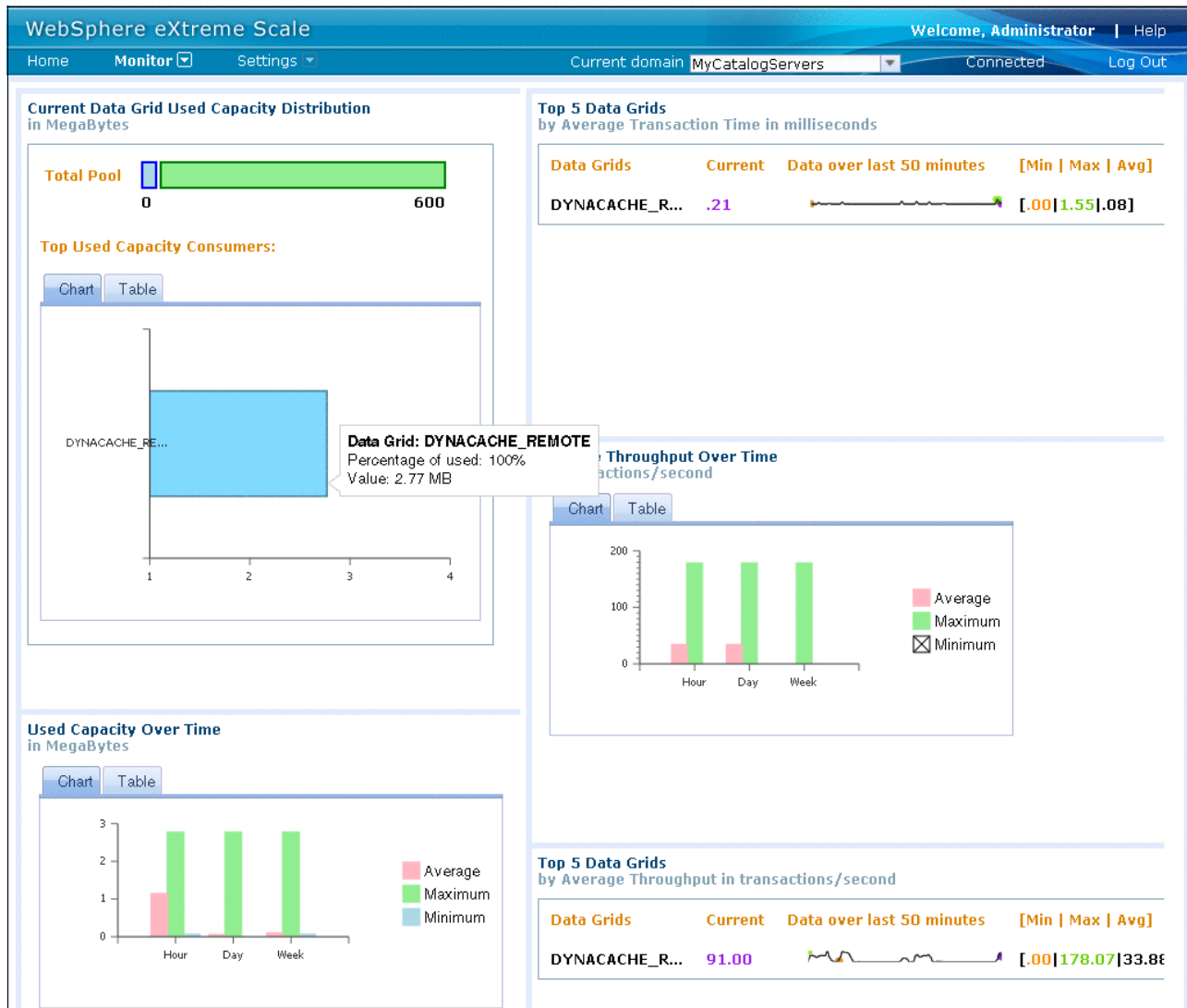


Figure 3-15 WebSphere eXtreme Scale web console - data grid overview

For further information about the web console, see the Information Center at:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/txsmonitoroversw.html>

3.8.2 Monitoring with xsadmin

WebSphere eXtreme Scale provides a command line tool to query the eXtreme Scale runtime deployment configuration. The **xsadmin** tool has the following uses:

- ▶ Verify that the grids are running
- ▶ Verify where partitions and replicas are running
- ▶ Query map sizes

The **xsadmin** command is run from \bin directory of the configuration. For example, when eXtreme Scale is installed on WebSphere Application Server, **xsadmin** can be started as follows:

WebSphere install\bin\xsadmin.bat

If you run **xsadmin** without any parameters, you are provided with a list of options for the operations you can perform and parameters for providing non-default connectivity options. Because you are running the catalog server in the deployment manager with default ports, you need to provide the **-dmgr** parameter. Useful commands are:

- ▶ To list running grids:
`xsadmin -dmgr -l`
- ▶ To show hosts and the location of all primary and replica partitions:
`xsadmin -dmgr -containers`
- ▶ To list running partitions with their corresponding sizes in terms of number of entries and used heap in KB:
`xsadmin -dmgr -mapsizes`

Further information about **xsadmin** can be found in the Information Center at:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/txsxsadmin.html>

3.8.3 Monitoring with the Tivoli Performance Viewer

The Tivoli Performance Viewer is the built-in monitoring console provided with the WebSphere administrative console. WebSphere administrators typically use it within the application server environment to monitor performance metrics (PMI metrics) for items such as memory usage and thread and connection pool utilization. WebSphere eXtreme Scale adds more PMI metrics into the same interface. These metrics provide more information about how eXtreme Scale is running within a given application server.

The Tivoli Performance Viewer can be used for obtaining eXtreme Scale statistics such as:

- ▶ Sizing: The number of entries and size of map
- ▶ Cache statistics: Includes cache retrievals and hit rates
- ▶ Performance: Retrieval response times

Tip: You can use PMI to monitor your environment only when you are using eXtreme Scale with WebSphere Application Server. If you have a stand-alone deployment of eXtreme Scale, then you need to use the eXtreme Scale web console (see Section 3.8.1, “Using the eXtreme Scale web console” on page 47).

By default, the Tivoli Performance Viewer does not show all PMI statistics. To see the performance statistics for the grid, you need to enable the ObjectGrid and ObjectGrid Maps modules. To do this, use the following procedure for each of the grid cluster members:

1. In the administrative console for the WebSphere process hosting the eXtreme Scale grid, click **Monitoring and Tuning** → **Performance Monitoring Infrastructure (PMI)**.
2. Click the grid container server, for example, `XS_GridCluster_Member1`.
3. Verify that **Enable Performance Monitoring Infrastructure (PMI)** is selected.
4. Click **Custom**.

- Click **ObjectGrids** on the left side. Select the ObjectGrid counters in the panel on the right and click **Enable**. The counter status changes to **Enabled**, as shown in Figure 3-16.

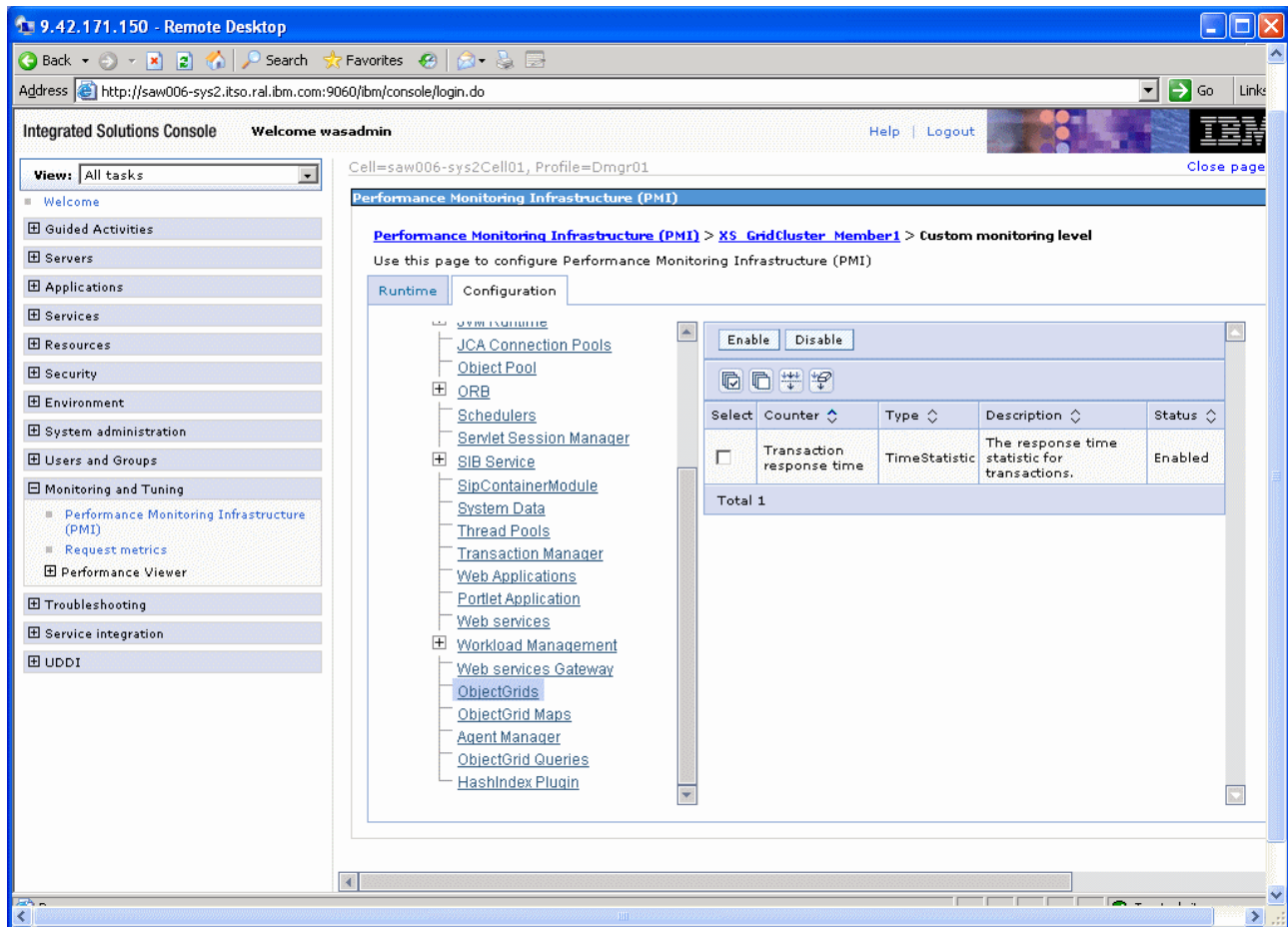


Figure 3-16 ObjectGrid PMI configuration

- Next, click **ObjectGrid Maps** on the left side. Select all of the ObjectGrid Map counters listed in the panel on the right and click **Enable**. The counter status will change to **Enabled**, as shown in Figure 3-17 on page 52.

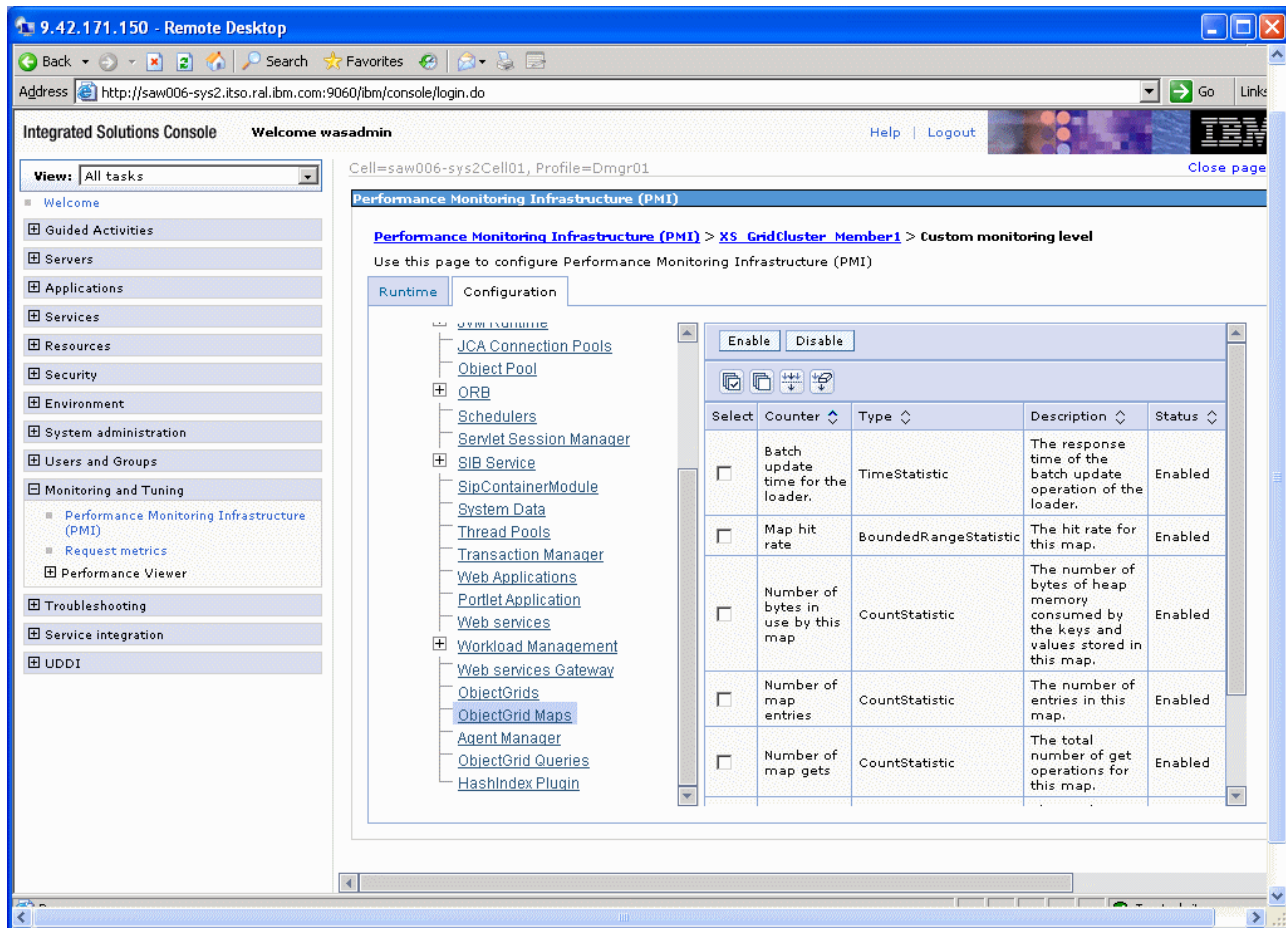


Figure 3-17 ObjectGrid Maps PMI configuration

7. Save your changes.
8. Synchronize the node and restart the grid cluster.

After the PMI statistics for eXtreme Scale have been enabled, you can see grid counters in the Tivoli Performance Viewer. Use the following procedure to view the grid counters.

1. In the administrative console, click **Monitoring and Tuning** → **Performance Viewer** → **Current Activity**.
2. Select the grid cluster servers that you want to monitor and click **Start Monitoring**.
3. Click a monitored server to open the Tivoli Performance Viewer for that server.
4. Expand **Performance Modules** on the left side and select the modules under ObjectGrids and ObjectGrid Maps that you want to view.
5. When the view refreshes, the performance statistics display, as shown in Figure 3-18 on page 53.

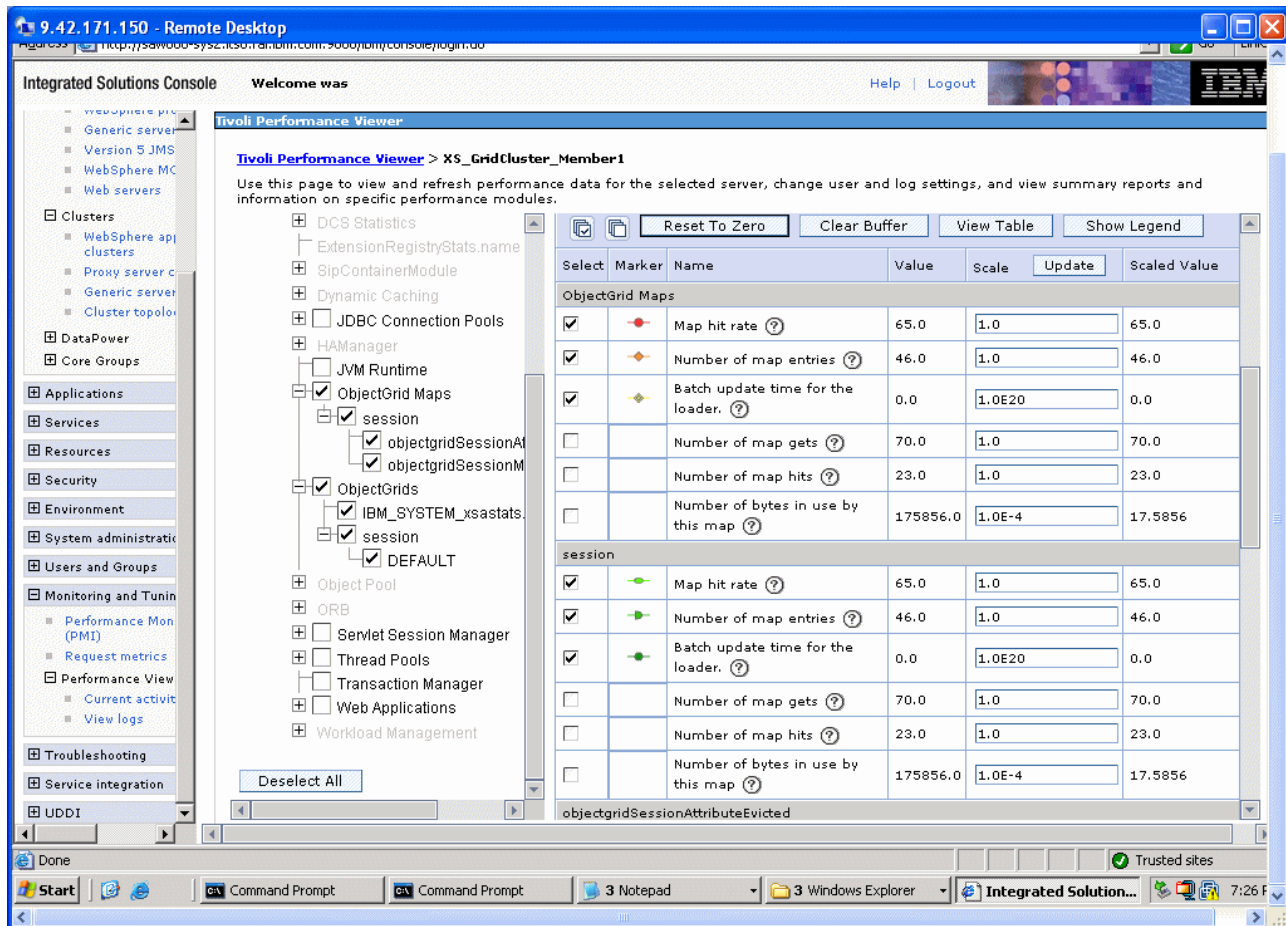


Figure 3-18 Object grid monitoring using the Tivoli Performance Viewer

3.8.4 Monitoring MBeans statistics with wsadmin

Managed beans (MBeans) can be used to track statistics in your eXtreme scale environment. When eXtreme Scale is deployed with WebSphere Application Server, the **wsadmin** tool can be used perform the following tasks:

- ▶ Get a reference to the eXtreme Scale MBeans using the AdminControl scripting object
- ▶ Invoke the exposed operations on eXtreme Scale MBeans using AdminControl scripting object

Example 3-1 shows a code sample that accesses managed beans to get statistics for a cache instance. The sample was run using the **wsadmin** tool.

Example 3-1 Statistics using MBean

```
nodeSpecification = ",node=" + nodeName
queryString = "type=DynaCache,process=" + serverName + nodeSpecification + ",*"
mbean = AdminControl.queryNames(queryString)
cacheInstances = AdminControl.invoke(mbean,"getCacheInstanceNames").split(linesep)
stats = AdminControl.invoke(mbean, "getAllCacheStatistics",
cacheInstance).split(linesep)
```

In the example the following variables were used:

- ▶ *nodeName* is the node name where application is deployed
- ▶ *serverName* is the sever name for which statistics are being collected

In our lab, we used a sample script called DynaCacheStatisticsCSV.py that collects cache statistics using MBeans and stores the data in a file as a comma-separated value (CSV) file. Example 3-2 shows how to use this script to collect cache statistics. The output of the script is a CSV file where data is written every ten seconds. We used baseCache as the cache instance, which is the default cache instance in WebSphere Application Server. You can modify this command according to your environment.

Obtaining the sample script: The DynaCacheStatisticsCSV.py script is available for download at:

<http://github.com/kelapure/dynacache/blob/master/scripts/DynaCacheStatisticsCSV.py>

Example 3-2 Dynacache statistics script

```
wsadmin -user wasadmin -password wasadmin -lang jython -f
DynaCacheStatisticsCSV.py server1 dyna_cache.log "-cacheInstance baseCache
-sleepUnit seconds -sleepInterval 10"
*sys-package-mgr*: processing modified jar, 'C:\Program
Files\IBM\WebSphere\AppServer\lib\wsobjectgrid.jar'
*sys-package-mgr*: processing modified jar, 'C:\Program
Files\IBM\WebSphere\AppServer\lib\wsogclient.jar'
WASX7209I: Connected to process "dmgr" on node deepkhanCellManager01 using SOAP
connector; The type of process is: DeploymentManager
WASX7303I: The following options are passed to the scripting environment and are
available as arguments that are stored in the argv variable: "[server1, dy
na_out.log, -cacheInstance baseCache -sleepUnit seconds -sleepInterval 10]"
['-cacheInstance', 'baseCache', '-sleepUnit', 'seconds', '-sleepInterval', '10']

DEBUG INFO:
    serverName= server1
        version=
        majorVersion=
        minorVersion=
    fileName= dyna_out.log
    nodeName= thinkNode01
    instanceName= baseCache
    sleepUnit= seconds
    sleepInterval= 10
        sleepMilliseconds= 10000
    fileAppend= false
instance: baseCache
```

The graphical presentation shown in Figure 3-19 on page 55 was created using the CSV file output using the graphing capability of a spreadsheet. It shows substantial increase in cache hits after the data is available as cache entries.

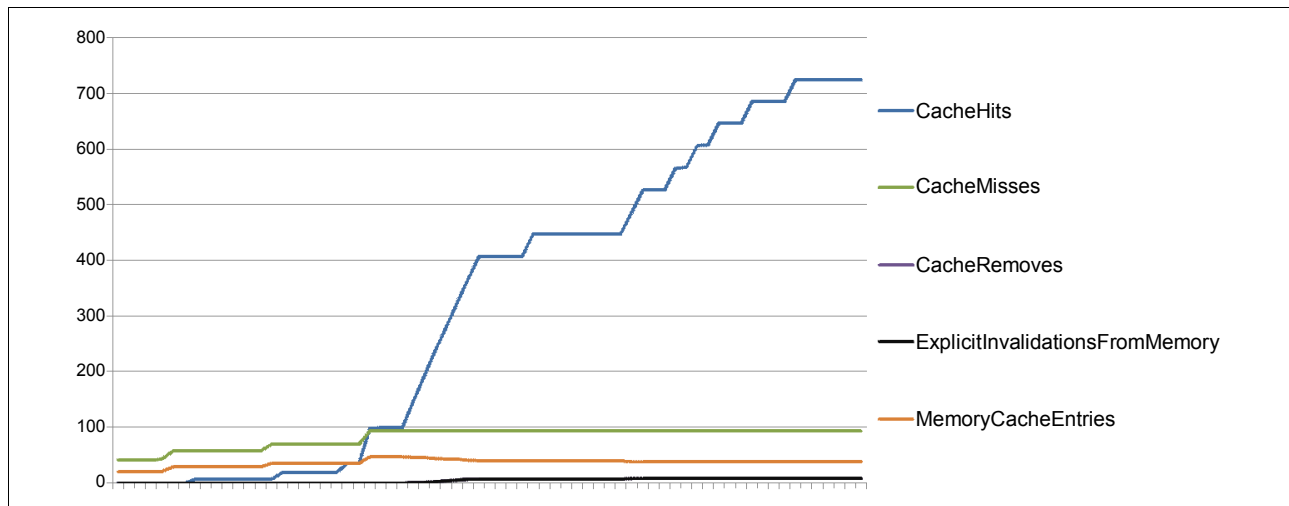


Figure 3-19 Graphical presentation of cache statistics

3.8.5 Monitoring MBean statistics in JConsole

If there is not enough detail in the Performance Viewer or **xsadmin**, it is worth having a look at the MBean statistics in JConsole. JConsole provides extensive and detailed metrics for grids, containers, and shards. JConsole can be run against WebSphere Application Server using the batch file provided in Example 3-3. Change the variables for HOST, BOOTSTRAP, and WAS_HOME to match your environment.

Example 3-3 Sample batch file for running JConsole against WebSphere Application Server

```
@ECHO OFF
set HOST=localhost
set BOOTSTRAP=9809
set WAS_HOME=C:\IBM\WebSphere\AppServer

set JAVA_HOME=%WAS_HOME%\java
echo Connecting to service:jmx:iiop://%HOST%:%BOOTSTRAP%/jndi/JMXConnector
set CLASSPATH=%JAVA_HOME%\lib\jconsole.jar;%JAVA_HOME%\lib\tools.jar;%WAS_HOME%
  \runtimes\com.ibm.ws.admin.client_7.0.0.jar
%JAVA_HOME%\bin\jconsole.exe -J-Djava.class.path=%CLASSPATH%
  service:jmx:iiop://%HOST%:%BOOTSTRAP%/jndi/JMXConnector
```

3.8.6 Monitoring with vendor tools

WebSphere eXtreme Scale can be monitored using several popular enterprise monitoring solutions. Plug-in agents are included for IBM Tivoli Monitoring (ITM) and Hyperic HQ, which monitor WebSphere eXtreme Scale using publicly accessible management beans. CA Wily Introscope uses Java method instrumentation to capture statistics.

IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale

The IBM Tivoli Enterprise Monitoring Agent is a feature-rich monitoring solution that you can use to monitor databases, operating systems, and servers in distributed and host environments. WebSphere eXtreme Scale includes a customized agent that you can use to introspect eXtreme Scale management beans. This solution works effectively for both stand-alone and WebSphere Application Server deployments.

CA Wily Introscope

CA Wily Introscope is a third-party management product that you can use to detect and diagnose performance problems in enterprise application environments. You can configure CA Wily Introscope to introspect selected portions of the eXtreme Scale process to quickly view and validate eXtreme Scale applications. CA Wily Introscope works effectively for both stand-alone and WebSphere Application Server deployments.

Hyperic HQ

Hyperic HQ is a third-party monitoring solution that is available at no cost as an open source solution or as an enterprise product. WebSphere eXtreme Scale includes a plug-in that allows Hyperic HQ agents to discover eXtreme Scale container servers and to report and aggregate statistics using eXtreme Scale management beans. You can use Hyperic HQ to monitor stand-alone eXtreme Scale deployments.

For more information about using CA Wily Introscope and Hyperic HQ, see Monitoring with vendor tools at the following address:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/cxsmonitorvendor.html>



Installing WebSphere eXtreme Scale

This chapter provides instructions for installing WebSphere eXtreme Scale. This chapter includes the following sections:

- ▶ 4.1, “Installing with an existing WebSphere Application Server product” on page 58
- ▶ 4.2, “Installing stand-alone WebSphere eXtreme Scale” on page 63

4.1 Installing with an existing WebSphere Application Server product

To install WebSphere eXtreme Scale on WebSphere Application Server related products, add eXtreme Scale related jars and executables to the underlying WebSphere Application Server. The installation procedure involves two primary steps:

- ▶ Installing eXtreme Scale binaries into the WebSphere Application Server directories
- ▶ Augmenting the existing profiles

Augmenting is used to add additional product features to a profile. To make the existing WebSphere Application Server profiles aware of eXtreme Scale, augment each profile. This augmentation can be done either during the installation process or after installation using Profile Management Tool (PMT).

The process outlined here applies to WebSphere Commerce Server and WebSphere Portal Server, which both use the underlying WebSphere Application Server engine. The only difference is that the profile names to be augmented differ.

4.1.1 Installing WebSphere eXtreme Scale

Start the installation from the eXtreme Scale root installation directory. Use the following procedure to install eXtreme Scale:

1. In the Installables directory, run **install.bat/sh**. Figure 4-1 shows the welcome page of the Installation Wizard. Click **Next**.

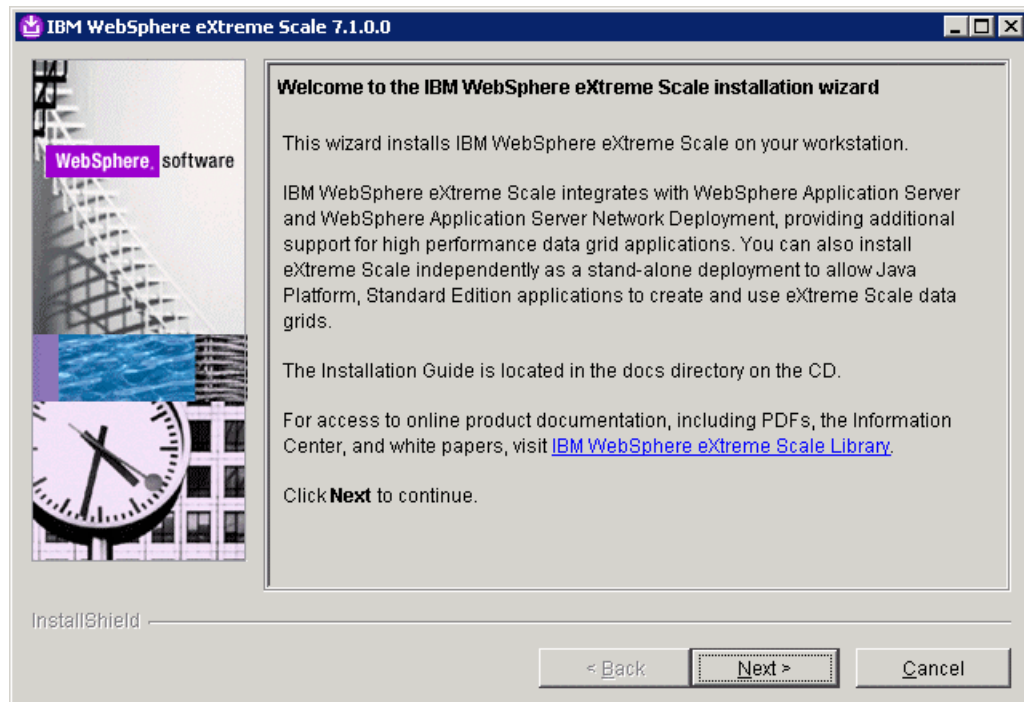


Figure 4-1 Installation Wizard welcome page

2. The next window shows the standard software license agreement page. Make sure that you have the necessary licenses in place before you install the software. Accept the agreement and then click **Next**.
3. Figure 4-2 shows the Installation directory page. Specify the installation directory of the base WebSphere Application Server. This step determines whether you install a stand-alone version or an integrated version of WebSphere eXtreme Scale. In this example, we are installing eXtreme Scale on an existing WebSphere Portal Server installation. Click **Next**.

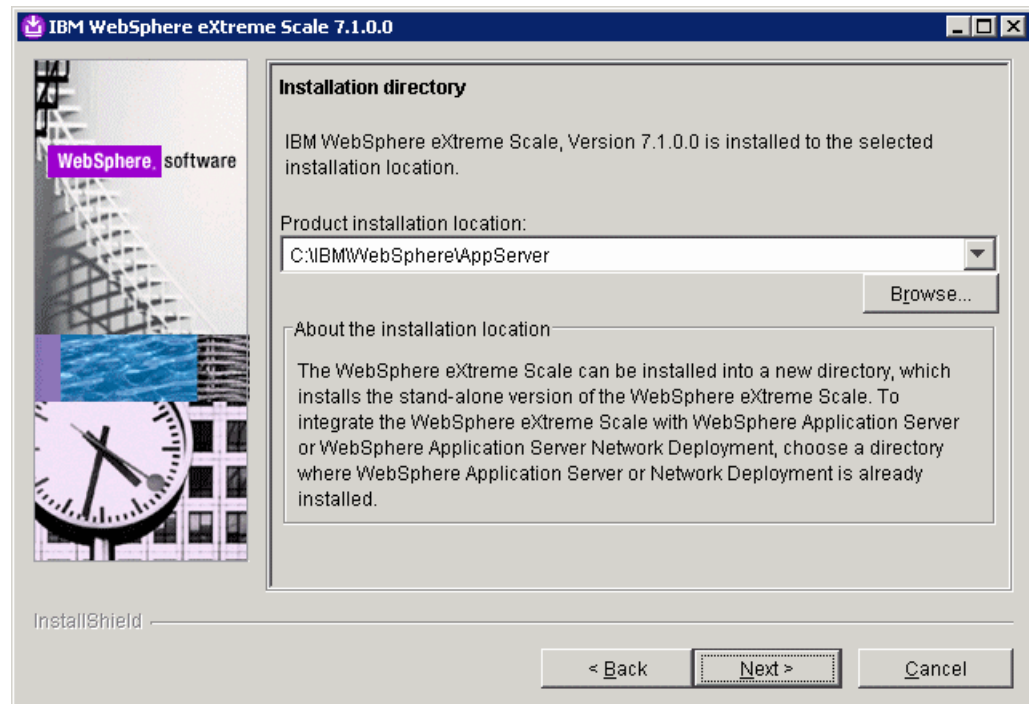


Figure 4-2 Specify the WebSphere Application Server home directory

4. Confirm that you are installing eXtreme Scale on an existing WebSphere Application Server Environment, as shown in Figure 4-3 on page 60. Because this is an integrated installation, confirm the selection by clicking **Next**.

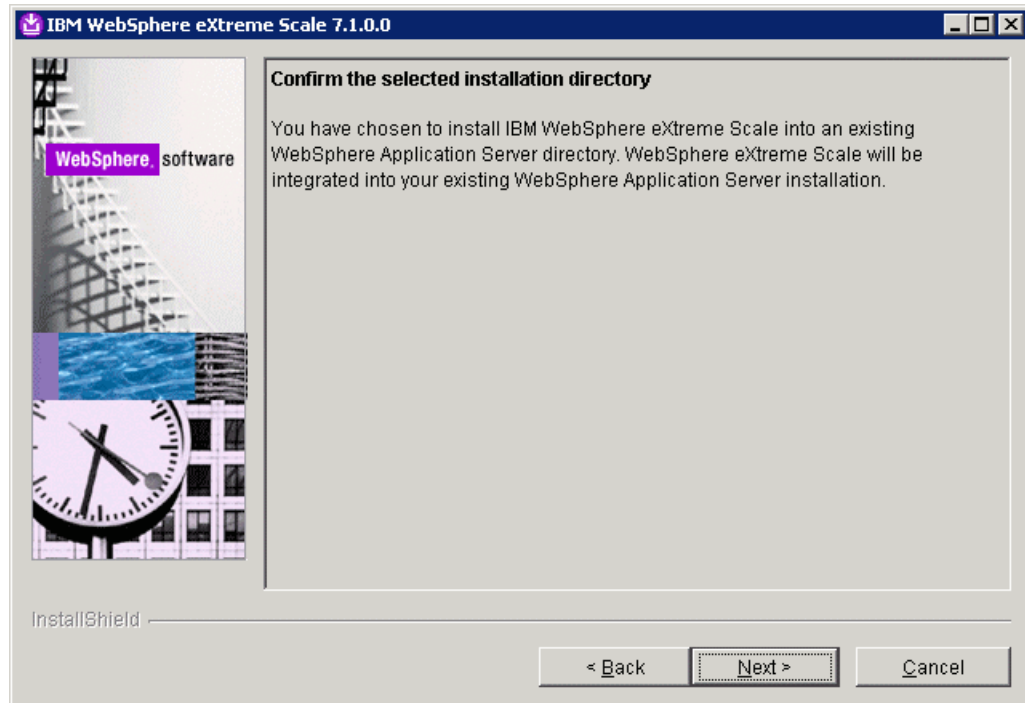


Figure 4-3 Confirm the installation directory

5. Figure 4-4 on page 61 shows the optional features you can select.

a. Select the type of installation:

- eXtreme Scale server installation

Performing a server installation makes the underlying WebSphere Application Server JVM act as a grid container. Select this option if you are planning to host the ObjectGrid maps in this JVM.

Choosing this option will automatically enable the client feature as well, as shown in Figure 4-4 on page 61.

- eXtreme Scale client installation

If you are planning to use this installation only as a client and not as a grid container, then choose the client installation. This type of installation reduces the memory footprint of the WebSphere Application JVM.

See 3.2, “Grid client and servers” on page 35 for more details.

The option you select will be determined by the role the node will play in the topology.

b. If your application uses either of the following deprecated APIs, select the appropriate API to install:

- Partition facility
- Stream query feature

In the Integration scenarios, we are not using any of the deprecated features. Details on the deprecated APIs are available in the Information Center at these websites:

Partitioning facility:

http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/topic/com.ibm.websphere.extremescale.wpf.doc/cwpfpartition_pdf.html

Stream query feature:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.javadoc.doc/topics/com.ibm.websphere.objectgrid/streamquery/package-summary.html>

Click **Next**.

Note: See Chapter 5, “Integrating WebSphere Portal with WebSphere eXtreme Scale” on page 69 and Chapter 6, “Integrating WebSphere Commerce with WebSphere eXtreme Scale” on page 99 to understand what features to choose when you integrate eXtreme Scale with the corresponding products.

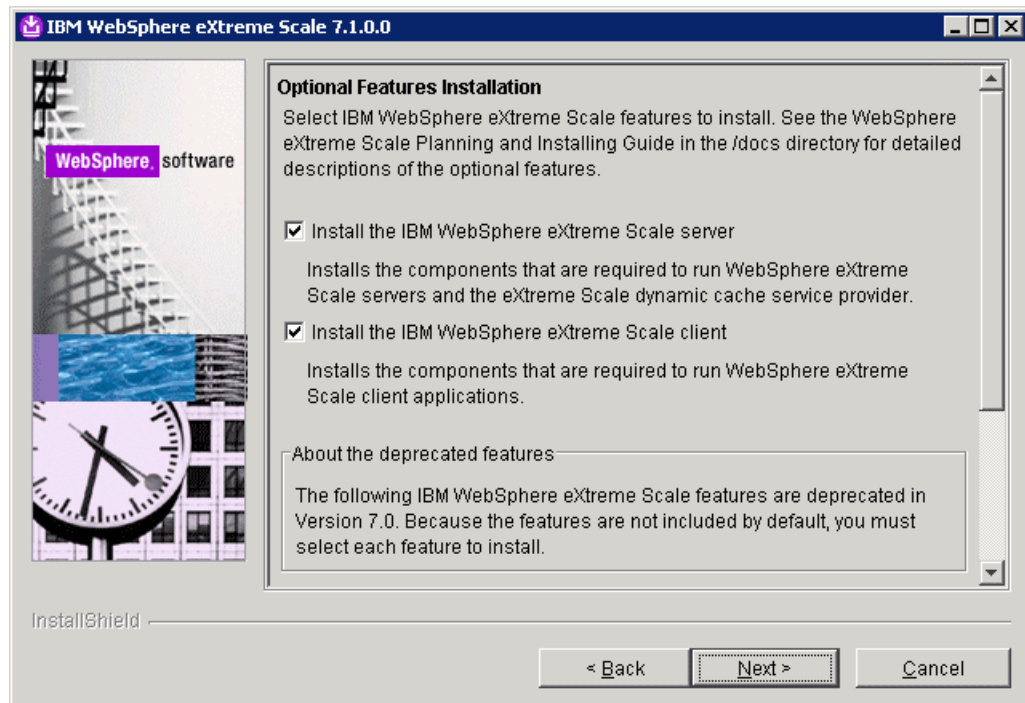


Figure 4-4 Optional features

- Figure 4-5 on page 62 shows the profile augmentation page. The Installation wizard automatically detects existing profiles in your installation and prompts you on whether you require an augmentation or not. Augmentation can also be performed after the product binary installation using PMT.

In this case, we are proceeding with the augmentation of our WebSphere Portal Server profile, wp_profile as part of the installation. If you have multiple profiles, the wizard allows you to augment all the profiles with a single click. Click **Next**.

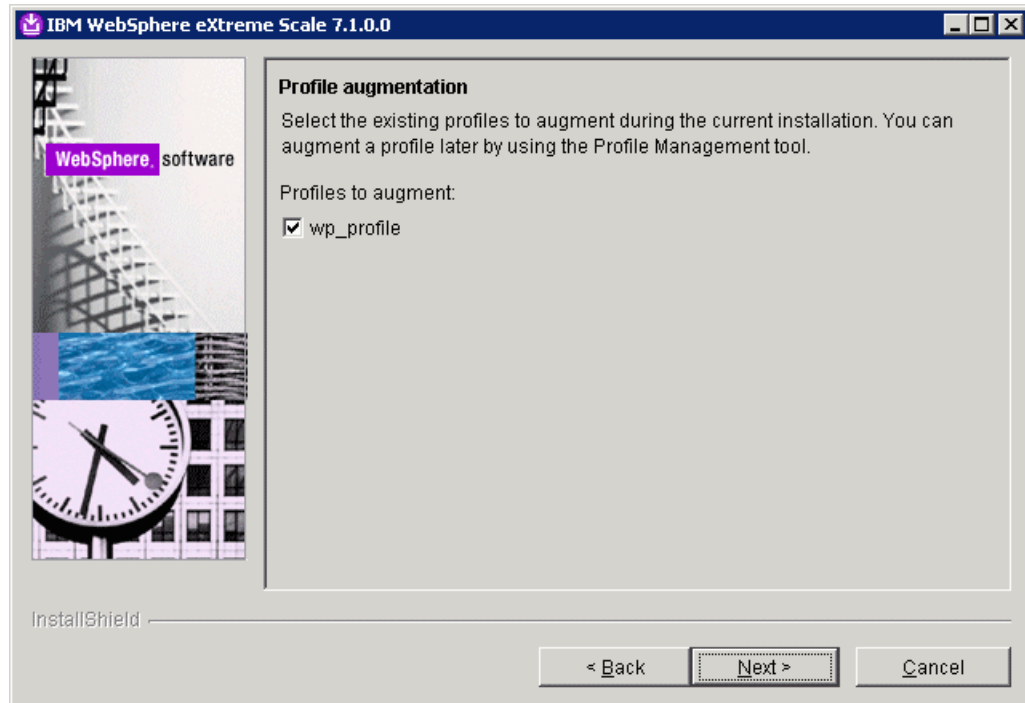


Figure 4-5 Profile augmentation

7. Figure 4-6 shows the summary page. Click **Back** if you need to revisit previous pages to change any of your selections. Click **Next** to complete the installation.

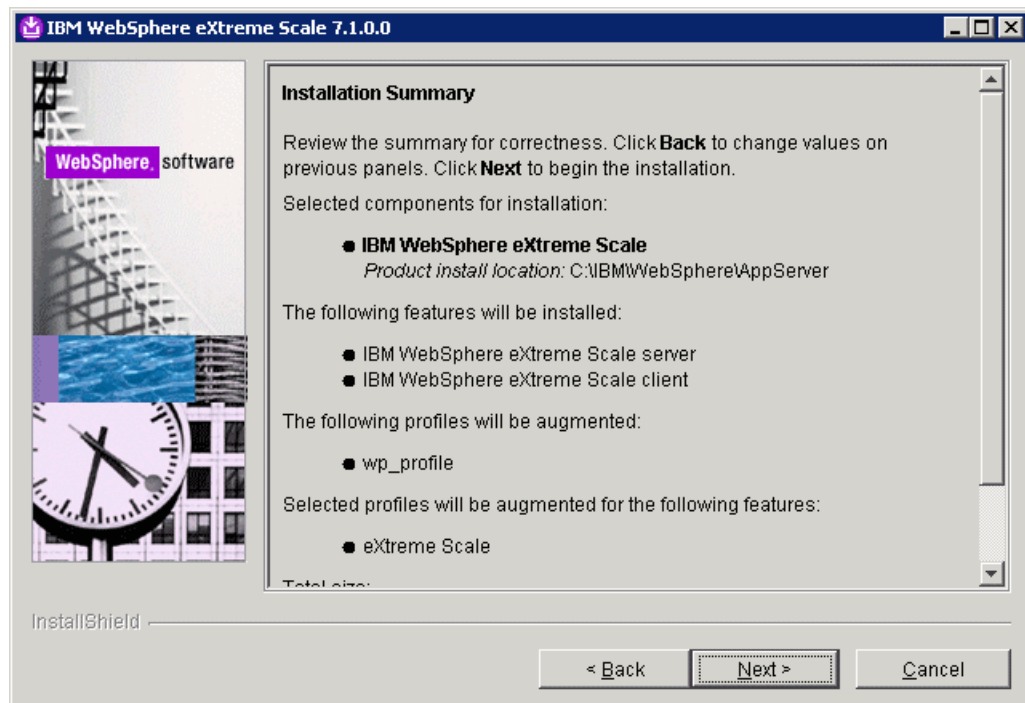


Figure 4-6 Installation Summary page

8. The next window shows the installation progress. When the installation is complete, click **Next**.

9. Figure 4-7 shows the final page of the process. It informs you of the success or failure of the installation. In the case of a failure, review the logs found in `WAS_HOME/logs/wxs/install` to find the reason for failure.

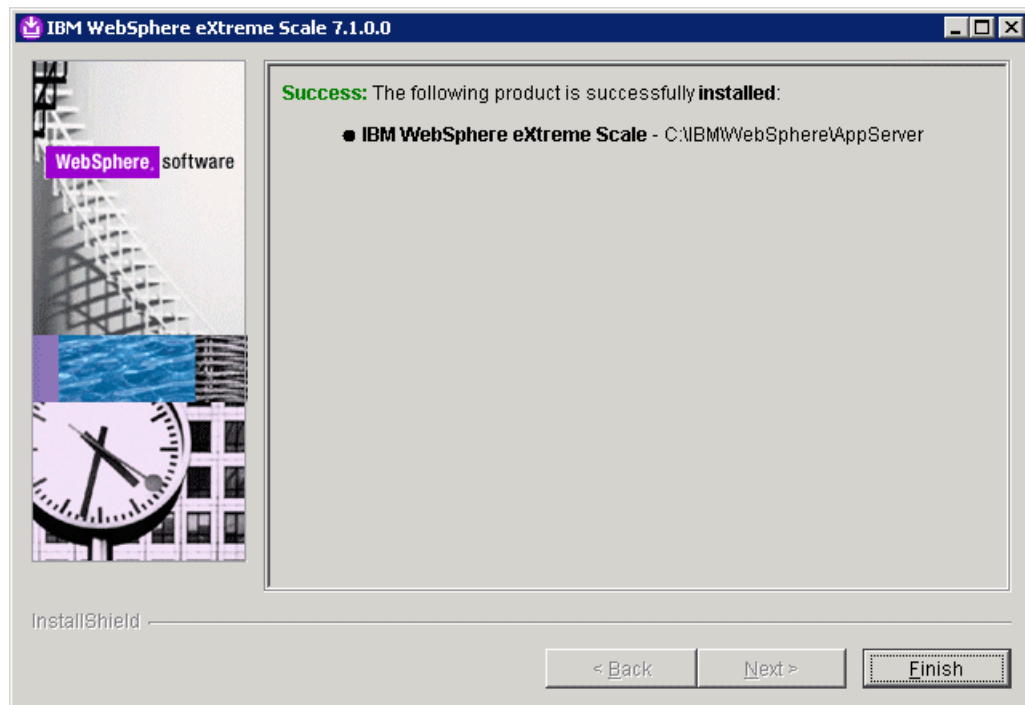


Figure 4-7 Installation final page

10. Click **Finish**.

4.1.2 Creating, starting and stopping catalog servers and containers

After augmenting the profiles, including the deployment manager profile, the deployment manager process, by default, becomes the catalog server. All other application server processes become the grid containers. These processes start by default when the deployment manager or the application server starts.

There are provisions by which you can specify the catalog server to run on any other Java process or even create a cluster of catalog servers through the administrative console of WebSphere Application Server. See Chapter 6, “Integrating WebSphere Commerce with WebSphere eXtreme Scale” on page 99 for more details.

4.2 Installing stand-alone WebSphere eXtreme Scale

Stand-alone installation of WebSphere eXtreme Scale follows the same general installation procedure described in 4.1, “Installing with an existing WebSphere Application Server product” on page 58. The differences between a stand-alone installation and an integrated installation of WebSphere eXtreme Scale are:

- The installation directory
On a stand-alone installation, make sure that the installation directory you specify is empty or does not exist.

- There are no profiles to augment

With the stand-alone installation, there are no profiles to augment.

- Console support

With the stand-alone installation, you can use WebSphere eXtreme Scale console support instead of using the WebSphere Application Server administrative console. The WebSphere eXtreme Scale Console provides support for monitoring the grid statistics. For more information about monitoring using the console, see 3.8.1, “Using the eXtreme Scale web console” on page 47.

A stand-alone installation of eXtreme Scale installs both the eXtreme Scale Client and Server into the machine. To install the client only, see:

<http://www-01.ibm.com/support/docview.wss?uid=swg24028267>

4.2.1 Installing a stand-alone environment

Use the following steps to install WebSphere eXtreme Scale in a stand-alone environment.

1. In the Installables directory, run `install.bat`. On the welcome page of the Installation Wizard, click **Next**.
2. Read and accept the agreement on the next page and then click **Next**.
3. Figure 4-8 shows the Installation directory page. Make sure that you select an empty directory or a non-existent directory. Henceforth we will refer to this directory as `OBJECTGRID_HOME`. Click **Next**.

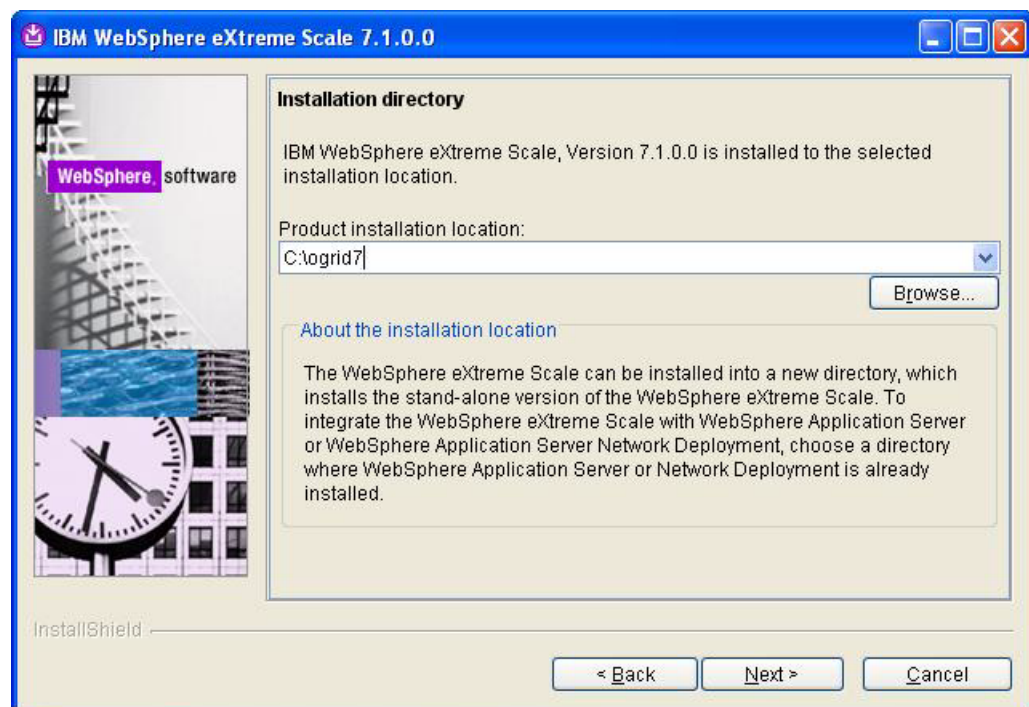


Figure 4-8 Installation directory page

4. Confirm that you are installing into an empty or new directory and click **Next**.
5. In the next window, shown in Figure 4-9 on page 65, select the WebSphere eXtreme Scale Console feature. This will place the `startConsoleServer` and `stopConsoleServer` executables in `OBJECTGRID_HOME/ObjectGrid/bin`. Click **Next**.

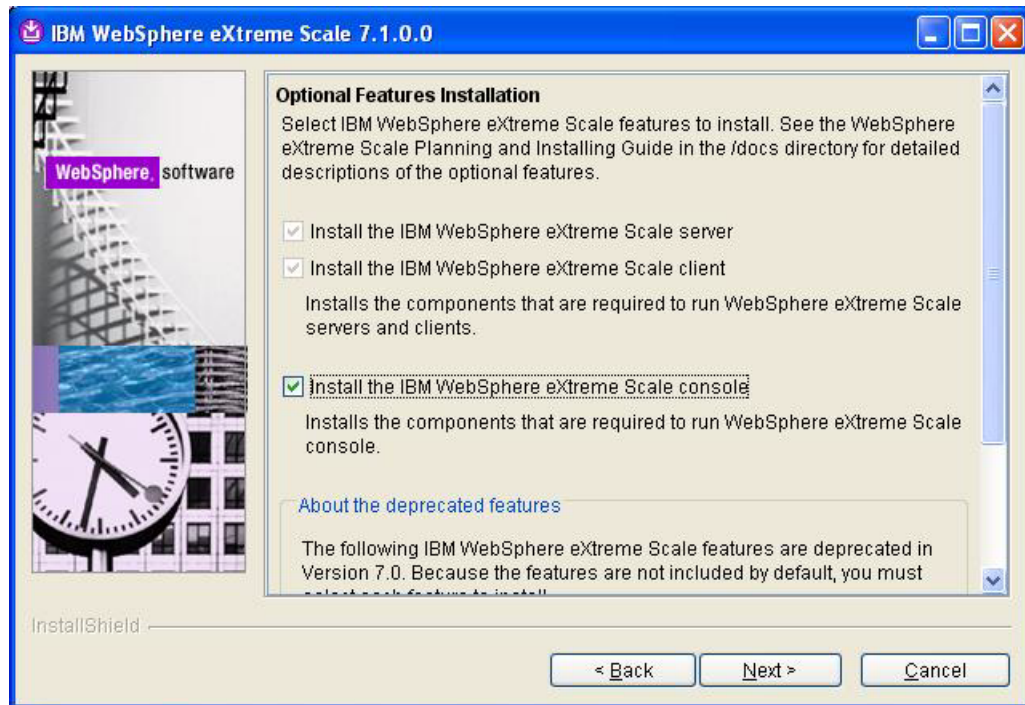


Figure 4-9 Optional features

6. The next window shows the installation progress. When the installation is complete, click **Next**.
7. Figure 4-10 shows the Installation Success page. In case of an installation failure, detailed logs and traces can be obtained from `OBJECTGRID_HOME/logs/wxs/install`.

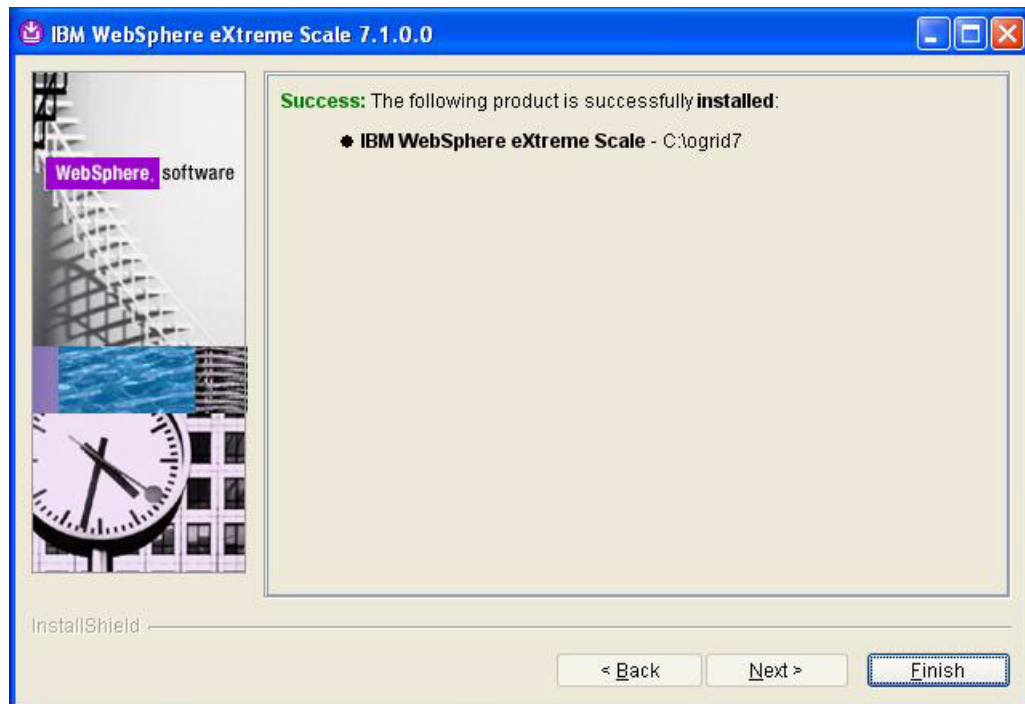


Figure 4-10 Installation Success page

8. Click **Finish**.

4.2.2 Starting a catalog server on a stand-alone installation

The stand-alone installation of WebSphere eXtreme Scale comes with an executable to help you start the catalog and grid servers.

Run **start0gServer** in the *OBJECTGRID_HOME*/ObjectGrid/bin directory with the following parameters:

```
start0gServer.bat/sh cat1 -catalogServiceEndpoints cat1:abc.com:6601:6602  
-listenerPort 2809
```

This command starts a catalog server on abc.com and listens at port 2809.

For information about starting catalog server clusters, refer to the following Information Center website for more details:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/txscatalogstart.html>

4.2.3 Stopping a catalog server

The following command stops the catalog server named cat1 that is listening on port 2809.

```
stop0gServer cat1 -catalogServiceEndpoints abc.com:2809
```

4.2.4 Starting grid containers on a stand-alone installation

Run the **start0gServer** command in the *OBJECTGRID_HOME*/ObjectGrid/bin directory with customized copies of the objectGrid and objectGridDeployment XML files. For example, the following command starts an **objectGridContainer** process using the Grid.xml and deployment.xml files to determine the configuration.

```
start0gServer.bat/sh c0 -objectGridFile Grid.xml -deploymentPolicyFile  
deployment.xml -catalogServiceEndpoints abc.com:2809
```

With this command, the grid server (c0) connects to the catalog server (abc.com) on port 2809. The catalog server needs to be listening on port 2809 in order for this grid container to start successfully.

The Grid.xml and deployment.xml file names used in the command are illustrative. The configuration will depend on the requirements for the application or product. WebSphere eXtreme Scale provides a getting started sample where you can review the configuration. These sample objectGrid.xml and objectGridDeployment.xml files are found in *OBJECTGRID_HOME*/ObjectGrid/gettingstarted/xml. You will see examples of the configuration files for eXtreme Scale in Chapter 5, “Integrating WebSphere Portal with WebSphere eXtreme Scale” on page 69 and Chapter 6, “Integrating WebSphere Commerce with WebSphere eXtreme Scale” on page 99.

4.2.5 Stopping the grid containers

The following command stops the grid container server named c0.

```
stop0gServer.bat c0 -catalogServiceEndpoints abc.com:2809
```

4.3 Applying maintenance

It is always a good idea to bring your software up to the latest fix level before implementing new solutions or applications. Maintenance for WebSphere products can be downloaded from the IBM Support Fix Central at:

<http://www-933.ibm.com/support/fixcentral/>

Specifically, for WebSphere eXtreme Scale Version 7.1 Client and Server instances, you will need Cumulative Fix 1. The latest fixes for WebSphere eXtreme Scale can be found at:

<http://www-01.ibm.com/support/docview.wss?uid=swg27018991#ver71>

The upgrade process is documented in the product Information Center in *Upgrading and migrating WebSphere eXtreme Scale Version 7.1* located at:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/txsupdate.html>



Integrating WebSphere Portal with WebSphere eXtreme Scale

In Section 2.4, “WebSphere Portal Server and WebSphere eXtreme Scale” on page 27, we talked about the challenges of a traditional approach towards session replication in a Portal architecture. We also explained how WebSphere eXtreme Scale can overcome the limitations of WebSphere’s data replication service and solve the availability and scalability challenges of a growing enterprise. In this chapter we will discuss the common use cases for introducing an eXtreme Scale solution into a Portal architecture. We will get into the technical details of setting up an eXtreme Scale and WebSphere Portal Server integration environment for session offload.

This chapter contains the following sections:

- ▶ 5.1, “When to use eXtreme Scale with Portal” on page 70
- ▶ 5.2, “Integration configuration and environment” on page 70
- ▶ 5.3, “Integration details” on page 73
- ▶ 5.4, “Monitoring” on page 92
- ▶ 5.5, “Guidelines and best practices” on page 95

5.1 When to use eXtreme Scale with Portal

WebSphere eXtreme Scale can serve as an efficient application state store for customer written portlets that make heavy use of session data. eXtreme Scale can be used to offload HTTP sessions to the grid which otherwise are stored and persisted either in memory or in a shared database.

Placing session data in an eXtreme Scale grid has the following benefits:

- ▶ Response times are reduced, increasing the throughput of the application
- ▶ Replication across the grid provides high availability
- ▶ Write-behind caching allows for more persistent storage, when required
- ▶ Moving session replication off of the database reduces load on the database
- ▶ If replicated sessions are currently being held in the JVM memory, then introducing eXtreme Scale frees up the JVM heap for other application data

Another advantage WebSphere eXtreme Scale offers is the zone-based support for data grids. This support provides high availability across physical locations by redundant placement of data. Traditionally, such enterprise computing environments were limited due to the performance constraints imposed by networks and real time data replication requirements. With zone support, only the metadata shared by the catalog servers is synchronously replicated. For details of zone support in eXtreme Scale, see 3.5, “Zone support” on page 39.

5.2 Integration configuration and environment

There are two ways in which eXtreme Scale can be hosted. It can either run on a stand-alone JVM or it can be installed in WebSphere Application Server. In this scenario, we use WebSphere Application Server to host eXtreme Scale. We have installed eXtreme Scale and augmented an existing WebSphere Application Server Network Deployment V7.0.0.11 profile. However, this is not a prerequisite for eXtreme Scale. eXtreme Scale can also run on a stand-alone JDK. See 3.7, “Common topologies” on page 41 for more details on possible eXtreme Scale topologies.

5.2.1 Prerequisites and requirements

This scenario assumes that you have the following items installed before starting the WebSphere Portal Server and WebSphere eXtreme Scale integration:

- ▶ An existing WebSphere Portal Server V7.0 stand-alone application server or clustered application server environment.
- ▶ WebSphere eXtreme Scale V7.1.0.1 (V7.1 with cumulative fix 1).
- ▶ A stand-alone JVM or WebSphere Application Server configuration to host the eXtreme Scale grid. In this scenario we use WebSphere Application Server.
- ▶ Serializable session objects for each of your custom portlets. Note that objects have to be serializable to place them in the grid containers.

5.2.2 Sample environment

We have configured a sample environment to demonstrate the HTTP session offload functionality provided by eXtreme Scale for a custom portlet. Figure 5-1 shows the deployment architecture of the sample environment. This is a remote topology where the eXtreme Scale grid is hosted on separate application servers than the Portal server. In this simple topology, both grid containers are on the same node. However, for high availability it is best to use separate nodes for grid containers. We have also created a high availability catalog service domain that hosts the catalog servers on the deployment manager and node agent.

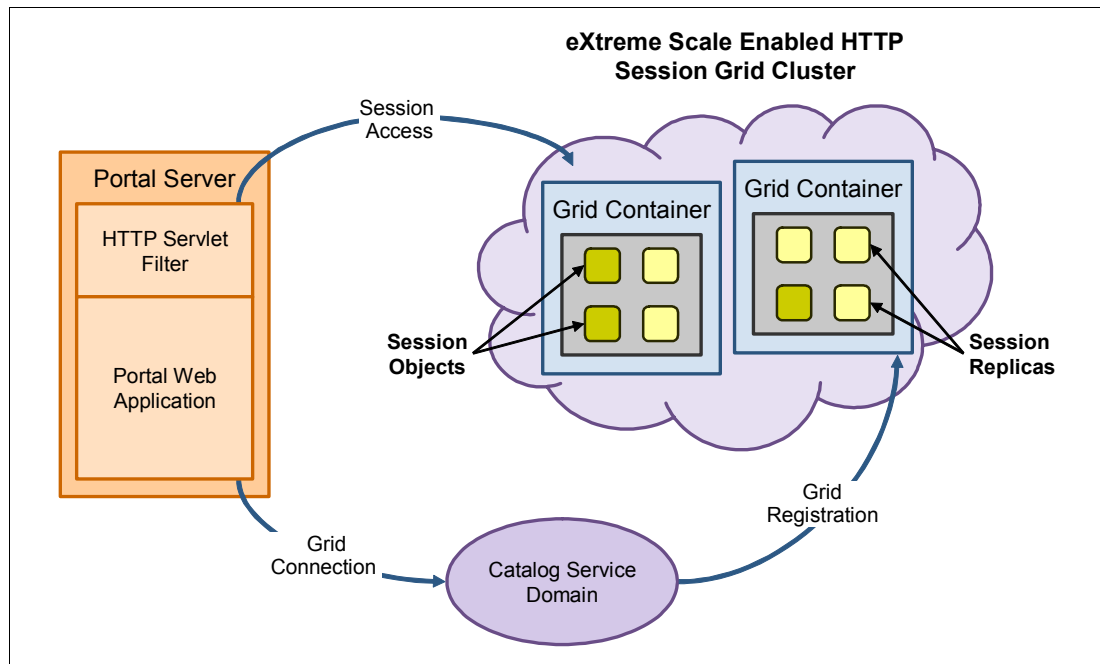


Figure 5-1 Sample eXtreme Scale - Portal integration architecture

5.2.3 Summary of the configuration

Figure 5-2 on page 72 shows the three broad categories of configuration required to build the sample eXtreme Scale and Portal integration environment:

- ▶ Catalog service domain
- ▶ eXtreme Scale grid cluster for session data
- ▶ WebSphere Portal Server configuration

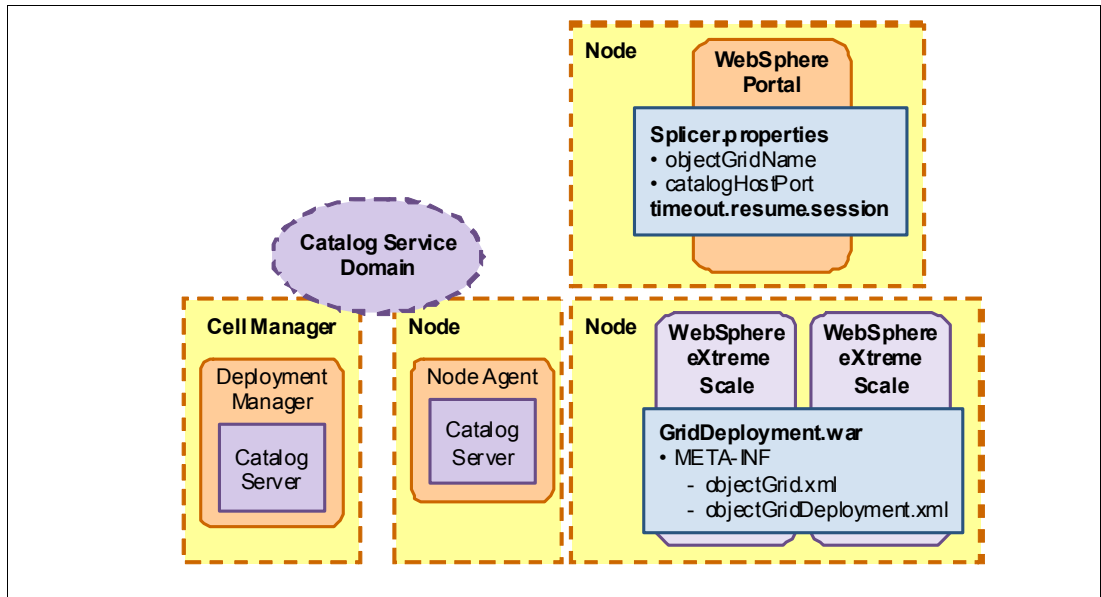


Figure 5-2 Sample environment topology

The following steps need to be performed before starting the integration:

1. Install WebSphere eXtreme Scale on all nodes and create/augment profiles.
2. Configure and cluster the catalog service domain.
3. Create eXtreme Scale session grid cluster and deploy the WAR file containing the grid XML files.
4. Configure the Portal server(s) properties to use the eXtreme Scale grid.
5. Start the environment in the following order: Catalog server, eXtreme Scale grid, and then Portal server.

We go over these steps in detail in the following sections.

5.2.4 Custom portlets

To demonstrate the use of eXtreme Scale for session persistence in a Portal environment, we use the custom Shopping Portlet. When the portlet is first called in a session, it adds 80 KB of data to the session. As shopping continues, more items are added to the Shopping Cart object in the session. Figure 5-3 on page 73 shows the custom Shopping Portlet.








JSR168 Shopping Portlet 0				
Welcome to the performance store.				
Check back later for specials!				
Large session size (80000) in use.				
Items for Sale			Your Shopping Cart	
	\$2300	Item #1 Intellistation M Pro [info] [purchase]	4	Intellistation POWER Series \$31600
	\$7900	Item #2 Intellistation POWER Series [info] [purchase]	Total: \$31600	
	\$1000	Item #3 NetVista A Series [info] [purchase]	[empty cart]	
	\$1100	Item #4 NetVista S Series [info] [purchase]		
	\$1200	Item #5 NetVista M Series [info] [purchase]		
	\$1300	Item #6 NetVista X Series [info] [purchase]		
	\$9000	Item #7 eServer pSeries 610 [info] [purchase]		

Figure 5-3 Shopping portlet

Important: The session objects in this portlet are made serializable to place them in the grid.

5.3 Integration details

The following steps need to be performed before integrating eXtreme Scale with an existing Portal setup for session management:

1. Install eXtreme Scale and augment the profiles.
2. Configure the eXtreme Scale and Portal Server environments:
 - a. Set up a catalog service domain.
 - b. Set up a clustered eXtreme Scale session grid.
 - c. Configure Portal Server.
3. Validate the configuration.

We will now get into details of the steps to create our sample environment.

5.3.1 Installing WebSphere eXtreme Scale for integration with Portal

In this section, we describe only those installation considerations specific to WebSphere Portal and the use of the eXtreme Scale session manager. The WebSphere eXtreme Scale installation is described in detail in Chapter 4, “Installing WebSphere eXtreme Scale” on page 57.

Because we are not going to discuss the installation of Portal Server, we are going to make the following assumptions about the installation environment:

- ▶ The WebSphere Portal Server installation has already been performed on WebSphere Application Server Network Deployment v7.0.0.11 (or any supported release).
- ▶ A Portal application server or cluster already exists that we are going to configure for use with eXtreme Scale.

At a high level, the installation steps required to prepare the Portal environment for eXtreme Scale session management are:

1. Install WebSphere Application Server Network Deployment on the eXtreme Scale node(s) with the appropriate patches.
2. Install WebSphere eXtreme Scale on top of the WebSphere Application Server installations with the appropriate patches.
3. Install WebSphere eXtreme Scale on top of the Portal Server installations with the appropriate patches.

Installation option: There are two installation options for WebSphere eXtreme Scale: Client and server installations. It is important to note that you need *server* installations for the eXtreme Scale nodes and a *client* installation for the Portal nodes.

4. Augment the existing Portal profiles to enable the WebSphere eXtreme Scale functionality. After eXtreme Scale has been installed on all of the nodes, the existing profiles need to be augmented with eXtreme Scale. For Portal, augment the Portal profile on all the nodes. On the WebSphere Application Server, augment the deployment manager and application server profiles.

Profile augmentation: You can select the profiles to augment while installing eXtreme Scale on WebSphere Application Server and Portal Server. You can also run the Profile Management Tool after the installation to augment the profiles.

For details on augmenting an existing profile, see Chapter 4, “Installing WebSphere eXtreme Scale” on page 57.

5. Create new WebSphere eXtreme Scale custom profiles for the WebSphere eXtreme Scale nodes and federate them to the deployment manager.

After you have completed these steps, you can configure the WebSphere eXtreme Scale Session management environment.

5.3.2 Configuring eXtreme Scale

This section tells you how to configure various components of eXtreme Scale. To set up the eXtreme Scale environment, do the following:

- ▶ Create a catalog service domain with clustered catalog servers.
- ▶ Define a clustered eXtreme Scale session grid.
- ▶ Configure the eXtreme Scale session grid.

The following sections tell you how to perform these steps.

Creating a catalog service domain

We will define the catalog service domain as shown in Figure 5-4. The catalog server is needed to manage the placement of eXtreme Scale partitions and to allow clients to connect to the grid.

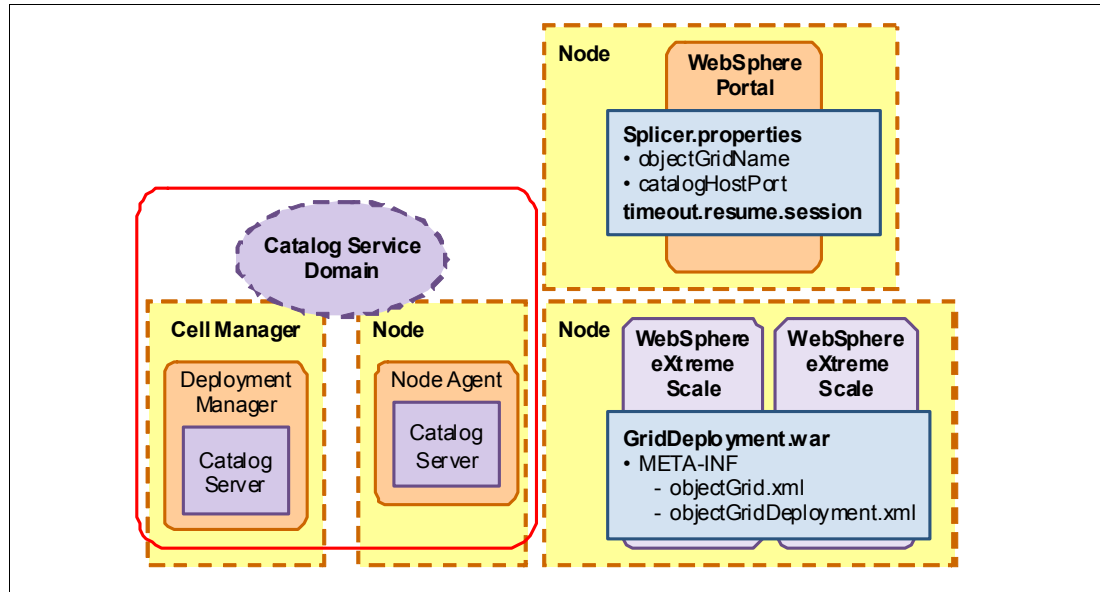


Figure 5-4 Configuring the catalog service domain

By default, the WebSphere eXtreme Scale catalog server will run in the deployment manager of the cell. This is a convenient default, but is not resilient if it is the only catalog server. The catalog server can be configured to run in other servers as well, such as the node agent(s) and application servers. Configure at least two or three catalog servers to run, ideally on separate physical servers. For production deployments, have at least three catalog servers.

We configured the catalog servers to run on the deployment manager and a single node agent by creating a catalog service domain. This is simply a grouping of local or remote catalog servers. When we define a catalog service domain of local catalog service endpoints we are doing two things:

- Declaring where the catalog services will run
- Providing the configuration so that eXtreme Scale grids and clients can connect to the eXtreme Scale catalog servers

Catalog server ports: When configuring the catalog server ports, you need to provide port numbers for a client port, listener port, and JMX port. What you provide varies depending on whether the catalog server is hosted in a WebSphere application server or is running stand-alone.

For catalog servers that are hosted in WebSphere, both the listener port and JMX port are the same. They are simply the bootstrap port of the process. They are, by default, 9809 for the deployment manager and 2809 for the node agent.

The client port is used for eXtreme Scale internal communication and must be unique.

Use the following procedure to configure a catalog service domain:

1. In the WebSphere administrative console, click **System Administration** → **WebSphere eXtreme Scale** → **Catalog service domains**.
2. Click **New**.
3. Enter a name for the catalog server domain, for example, Clustered Catalog Servers.
4. Select **Enable this catalog service domain as the default**.
5. Add the catalog servers to the catalog service domain:
 - a. To add a catalog server to the deployment manager:
 - i. A catalog service endpoint will already exist. Select the deployment manager in the **Existing application server** drop down.
 - ii. Enter a client port value: 6600
 - iii. Enter a JMX port value: 9809

When hosting eXtreme Scale catalog servers in WebSphere Application Server, this port is the server's bootstrap address, which is 9809 by default on the deployment manager.
 - b. To add a catalog server to a node agent:
 - i. Click **New**.
 - ii. Select a node agent from the list in the Existing application server drop down.
 - iii. Enter a client port value: 6601

In the sample environment, the node agent is on the same hardware as the deployment manager, which means that we need separate ports to make them unique.
 - iv. Enter a JMX port value: 2809

Figure 5-5 on page 77 shows the configuration window being used.

General Properties

Name
Clustered Catalog Servers

☒ Enable this catalog service domain as the default unless another catalog service domain is explicitly specified.

JMX authentication credentials

Provide credentials for Java Management Extensions (JMX) authentication for retrieving status from the catalog server endpoints.

User ID

Password

Catalog Servers

New Delete

Select	Catalog Server Endpoint	Client Port	Listener Port	JMX Port
<input type="checkbox"/>	<input checked="" type="radio"/> Existing application server <input type="text" value="thinkCell01\thinkCellManager01\dmgr"/>	6600	<input type="text"/>	9809
	<input type="radio"/> Remote server <input type="text"/>			
<input type="checkbox"/>	<input checked="" type="radio"/> Existing application server <input type="text" value="thinkCell01\thinkNode02\nodeagent"/>	6601	<input type="text"/>	2809
	<input type="radio"/> Remote server <input type="text"/>			

Apply OK Reset Cancel

Figure 5-5 Create a catalog service domain

c. Click **Apply**.

At this point, the Listener Port field is disabled when adding existing servers. This is because WebSphere assumes the bootstrap port for the servers we are configuring and so we cannot change them.

At the time of this writing, if you are *not* using the default bootstrap ports (9809 for deployment managers and 2809 for node agents), these need to be added to the **Listener Port** field. Otherwise the grid will attempt to use the defaults. Even if you are using the defaults, you can choose to do this to add clarity in the configuration.

- i. In the new catalog service domain, select the two catalog server entries and click **Edit**.
- ii. Enter the listener port values appropriate to your environment. For our example environment, they are:
 dmgr Listener Port = 9809
 nodeagent Listener Port = 2809

The fully configured catalog service domain is shown in Figure 5-6 on page 78.

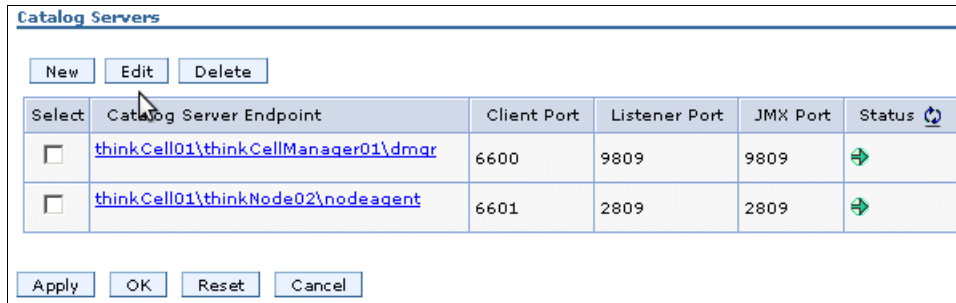


Figure 5-6 The fully configured Catalog service domain (shown after restart)

6. Save the changes to the master configuration.
7. After changing catalog server settings, restart the deployment manager and node agent servers that are to host the catalog servers.

Important: All catalog servers need to be started at approximately the same time because they communicate with each other while starting.

You can verify that the catalog server has started correctly by checking the SystemOut.log files for the deployment manager and node agent. The files contain a message similar to that found in Example 5-1 indicating that the catalog server is starting.

Example 5-1 Log message to indicate that the catalog server is starting

```
[10/21/10 14:24:52:296 CDT] 0000000c ServerImpl I CW0BJ2518I: Launching
ObjectGrid catalog service: thinkCell01\thinkNode02\nodeagent.
```

After the deployment manager and node agent are started, they will form a catalog service cluster (one master catalog and one standby) and a message similar to Example 5-2 displays.

Example 5-2 Log message to provide the status of the catalog server cluster

```
[10/21/10 14:25:37:343 CDT] 00000012 CatalogServer I CW0BJ8109I: Updated catalog
service cluster CatalogCluster[thinkCell01, 1 master: 1 standbys] from server
thinkCell01\thinkNode02\nodeagent with entry CatalogServerEntry...
```

Important: Because the deployment manager now hosts a catalog server, it is an important part of the runtime infrastructure. Prior to use with WebSphere eXtreme Scale, you were able to stop and start the deployment manager at will because it was not needed for the application servers. This is no longer the case and stopping the deployment manager will reduce the catalog server availability, and therefore site availability.

After the catalog service domain is configured in the eXtreme Scale cluster, we need to make the Portal servers aware of the catalog cluster. We will do this when we configure WebSphere Portal Server. We will configure the catalogHostPort in the splicer.properties file using the format `dmgrHost:listenerPort,nodeagentHost:listenerPort`.

Creating a clustered eXtreme Scale session grid

We will create a cluster of two application servers to host our session grid, as shown in Figure 5-7 on page 79. We are creating this cluster with only two members for simplicity and

to demonstrate a clustered grid setup. However, the number of servers you will need to host the grid depends on the amount of session data your Portal application stores.

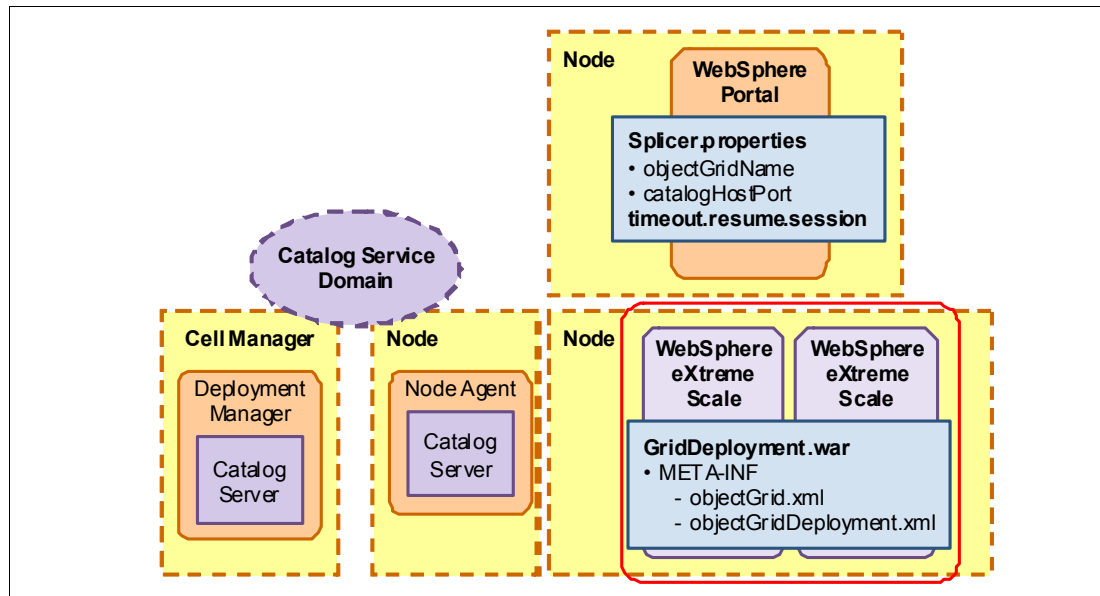


Figure 5-7 Setting up a clustered eXtreme Scale session grid

Use the following procedure to set up the grid cluster.

1. In the administrative console, click **Servers** → **Clusters** → **WebSphere application server clusters**. Click **New**.
2. Enter a cluster name, for example, XS Session Grid Cluster. Click **Next**.
3. Add the first cluster member with the appropriate details.
 - a. Member name: XS_GridCluster_Member1.
 - b. Select the node you want the servers to run on.
 - c. Click **Next**.
4. Create additional cluster members. For our sample environment, we will add only one additional member to make it a two member cluster.
 - a. Member name: XS_GridCluster_Member2.
 - b. Click **Add Member**.
 - c. Click **Next**.
5. Review the Summary and click **Finish**. The summary for our sample environment is shown in Figure 5-8 on page 80.

Create a new cluster

Step 1: Enter basic cluster information

Step 2: Create first cluster member

Step 3: Create additional cluster members

→ Step 4: Summary

Summary

Summary of actions:

Options	Values
Cluster Name	XS Session Grid Cluster
Core Group	DefaultCoreGroup
Node group	DefaultNodeGroup
Prefer local	true
Configure HTTP session memory-to-memory replication	false
Server name	XS_GridCluster_Member1
Node	saw006-sys2Node01(ND 7.0.0.11 WXS 7.1.0.0)
Weight	2
Clone Template	default
Clone Basis	Create the member using an application server template.
Generate unique HTTP ports	true
Server name	XS_GridCluster_Member2
Node	saw006-sys2Node01(ND 7.0.0.11 WXS 7.1.0.0)
Weight	2
Clone Template	Version 7 member template
Generate unique HTTP ports	true

Previous

Finish

Cancel

Figure 5-8 Session Grid Cluster summary

6. Save your changes.

Next, configure the eXtreme Scale session grid. To set up the session grid, you will need to package the `objectGridStandAlone.xml` and `objectGridDeploymentStandAlone.xml` files into an EAR file and deploy it on the WebSphere Application Server. These files define the grid and its deployment configuration. The “stand-alone” files are used for the configuration because they define a remote grid for HTTP sessions, which is the topology we are deploying.

1. Locate the `objectGridStandAlone.xml` and `objectGridDeploymentStandAlone.xml` files in the following directory:
`WAS_HOME\optionalLibraries\ObjectGrid\session\samples`
2. Create a temporary folder with a META-INF folder inside it. Place these files in the META-INF folder.
3. Modify the configuration files as follows:
 - `objectGridStandAlone.xml` contains the grid definition required by eXtreme Scale to host session data. This file by default will look like Example 5-3 and does not require any editing.

Example 5-3 Sample `objectGridStandAlone.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
```



```

<objectGrids>
  <objectGrid name="session">
    <bean id="ObjectGridEventListener"
      className="com.ibm.ws.xs.sessionmanager.SessionHandleManager"/>
    <backingMap name="objectgridSessionMetadata"
      pluginCollectionRef="objectgridSessionMetadata"
      readOnly="false"
      lockStrategy="PESSIMISTIC"
      ttlEvictorType="LAST_ACCESS_TIME"
      timeToLive="3600"
      copyMode="COPY_TO_BYTES"/>
    <backingMap name="objectgridSessionAttribute.*"
      template="true"
      readOnly="false"
      lockStrategy="PESSIMISTIC"
      ttlEvictorType="NONE"
      copyMode="COPY_TO_BYTES"/>
    <backingMap name="objectgridSessionTTL.*"
      template="true"
      readOnly="false"
      lockStrategy="PESSIMISTIC"
      ttlEvictorType="LAST_ACCESS_TIME"
      timeToLive="3600"
      copyMode="COPY_TO_BYTES"/>
  </objectGrid>
</objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="objectgridSessionMetadata">
      <bean id="MapEventListener"
        className="com.ibm.ws.xs.sessionmanager.MetadataMapListener"/>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>

```

-
- objectGridDeploymentStandAlone.xml shown in Example 5-4 contains the deployment configuration that eXtreme Scale will use to create the grids. You can use this with default values. However, it is advised that you review the contents.

Example 5-4 Sample objectGridDeploymentStandAlone.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="session">
    <mapSet name="sessionMapSet"
      numberOfPartitions="5"
      minSyncReplicas="0"
      maxSyncReplicas="0"
      maxAsyncReplicas="1"
      developmentMode="true"
      numInitialContainers="2"
      placementStrategy="PER_CONTAINER">

```

```

        <map ref="objectgridSessionMetadata"/>
        <map ref="objectgridSessionAttribute.*"/>
        <map ref="objectgridSessionTTL.*"/>
    </mapSet>
</objectgridDeployment>
</deploymentPolicy>

```

Revise the following parameters and change them as appropriate.

- numInitialContainers

This should be equal to the number of application servers in the topology. However, if the number of initial containers is lower than the number of application servers, it will lead to an imbalanced grid. The placement will not start till these many containers are started.

- numberOfPartitions

The default value is 5, which means that each container instance will start five new primaries, with its replicas spread across the other servers. The higher the value of numberOfPartitions, the more the work is spread out in event of a failover. The default works in most cases.

- developmentMode

This attribute influences where a shard is placed in relation to its peer shards. When set to false, (the default) a replica shard will not be placed on the same physical node as the primary. When set to true, shards from the same partition can be placed on the same physical machine. In either case, no two shards from the same partition are ever placed in the same container.

For the purpose of this sample environment, because we have both grid containers on the same physical node, we have to set the developmentMode to true to be able to see replicas placed in our grid.

- placementStrategy

There are two placement strategies.

The default strategy is FIXED_PARTITION, where the number of primary shards that are placed across available containers is equal to the number of partitions defined. In a single-container configuration this will be the total number of shards. In a multiple-container configuration, the total number of shards will be a multiple of this number, depending upon how many replicas are configured.

The alternate strategy is PER_CONTAINER, where the number of primary shards that are placed on each container is equal to the number of partitions that are defined, with an equal number of replicas placed on other containers.

We selected PER_CONTAINER for our sample session topology to support a multi-datacenter environment where clients in one data center prefer the primary partitions that are local to that data center. A standard FIXED_PARTITIONS strategy cannot guarantee the client will always talk to a primary in its own data center.

4. Rename objectGridStandAlone.xml to objectGrid.xml and objectGridDeploymentStandAlone.xml to objectGridDeployment.xml.

Note: Because we are running the containers in WebSphere Application Server, we must rename the *StandAlone.xml files to objectGrid.xml and objectGridDeployment.xml. The XML files must be a specific name for the grid container embedded in WebSphere Application Server to start.

5. Create a web archive file (WAR) with the objectGrid.xml and objectGridDeployment.xml files. The WAR file can be created by running the **jar** command from the temporary directory containing the META-INF folder. The jar command can be found in the *WAS_HOME/java/bin* directory. Example 5-5 shows an example.

Example 5-5 Creating Deployment xml WAR file

```
jar -cvf GridDeployment.war *
```

```
added manifest
ignoring entry META-INF/
adding: META-INF/objectGrid.xml(in = 1263) (out= 464)(deflated 63%)
adding: META-INF/objectGridDeployment.xml(in = 712) (out= 340)(deflated 52%)
```

6. Deploy the WAR file on the application server using the following steps:
 - a. On the WebSphere administrative console, click **Applications** → **New Application** → **New Enterprise Application**.
 - b. Click **Browse for the Local file system** and select the GridDeployment.war file.
 - c. Click **Next**.
 - d. On Preparing for the application installation, leave the default as Fast Path and click **Next**.
 - e. For the remainder of the installation, use all of the defaults except for Step 2: Map modules to servers, which prompts you to select the GridDeployment.war module. Instead, select XS Session Grid Cluster and click **Apply**, as shown in Figure 5-9 on page 84.

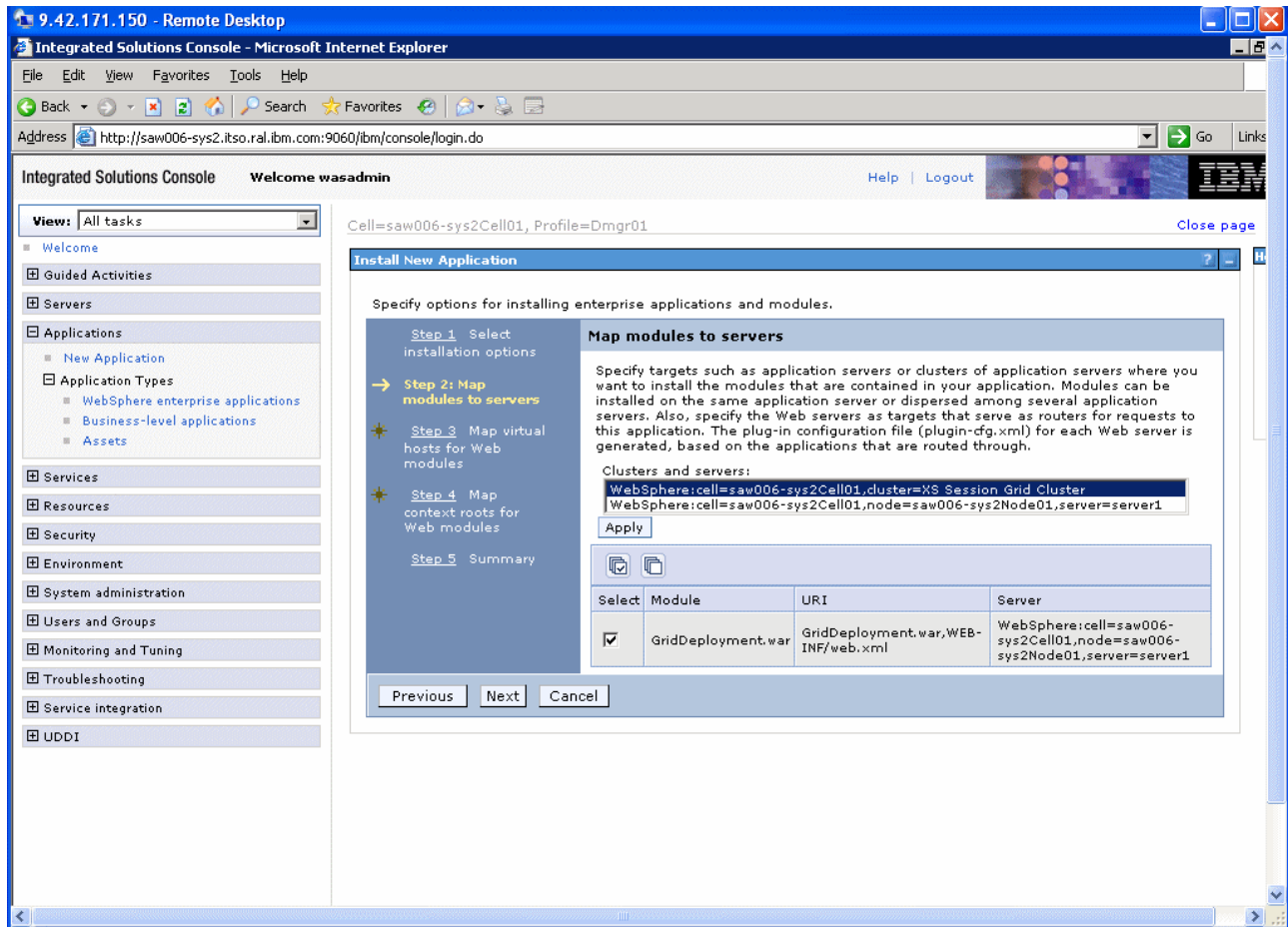


Figure 5-9 Deploying the Grid configuration

- f. On the Summary page, click **Finish** and save your changes.

Now, start the grid and verify the grid configuration.

1. To start the eXtreme Scale Grid using the administrative console, click **Servers** → **Clusters** → **WebSphere Application Server Clusters**.
2. Select **XS Session Grid Cluster** and click **Start**.
3. When the cluster starts, all of the cluster members start. Verify the grid configuration by looking at the `SystemOut.log` files. You will come across grid related log entries in the following order:
 - a. WebSphere eXtreme Scale Version information, as shown in Example 5-6.

Example 5-6 Log entries showing eXtreme Scale version

```
[11/2/10 23:39:36:152 EDT] 00000000 RuntimeInfo I CW0BJ0903I: The
internal version of WebSphere eXtreme Scale is v4.0 (7.1.0.0) WXS7.1.0.XS
[a1025.57350].
```

Note: The version information displayed in this message will reflect the fixpacks installed on your setup.

- b. Web module and grid startup, as shown in Example 5-7 on page 85.

Example 5-7 Log entries showing Grid startup

```
[11/2/10 23:39:49:402 EDT] 0000001b PeerManagerSe I   CWOBJ7700I: Peer
Manager service started successfully in server
(com.ibm.ws.objectgrid.leader.PeerManager@1c3d1c3d) with core group
(DefaultCoreGroup).
[11/2/10 23:39:50:059 EDT] 00000018 PeerManager   I   CWOBJ8601I:
PeerManager found peers of size 2
[11/2/10 23:39:50:496 EDT] 0000001b ServerImpl    I   CWOBJ8000I:
Registration is successful with zone (DefaultZone) and coregroup of
(saw006-sys2Cell01DefaultCoreGroup).
[11/2/10 23:39:50:512 EDT] 0000001b ServerImpl    I   CWOBJ1001I: ObjectGrid
Server saw006-sys2Cell01\saw006-sys2Node01\XS_GridCluster_Member1 is ready
to process requests.
[11/2/10 23:39:50:715 EDT] 0000001b XmlObjectGrid I   CWOBJ4701I: Template
map info_server.* is configured in ObjectGrid IBM_SYSTEM_xsastats.server.
[11/2/10 23:39:50:887 EDT] 0000001b XmlObjectGrid I   CWOBJ4701I: Template
map objectgridSessionAttribute.* is configured in ObjectGrid session.
[11/2/10 23:39:50:887 EDT] 0000001b XmlObjectGrid I   CWOBJ4701I: Template
map objectgridSessionTTL.* is configured in ObjectGrid session.
[11/2/10 23:39:51:559 EDT] 0000001b webapp        I
com.ibm.ws.webcontainer.webapp.WebGroupImpl WebGroup SRVE0169I: Loading Web
Module: GridDeployment.war.
```

- c. eXtreme Scale partitions and replicas activating, as shown in Example 5-8.

Example 5-8 Log entries showing Partition and Replica activation

```
[11/2/10 23:39:52:356 EDT] 00000026 ReplicatedPar I   CWOBJ1511I:
IBM_SYSTEM_xsastats.server:IBM_SYSTEM_ENTITYMANAGER_MAPSET:0 (primary) is
open for business.
[11/2/10 23:39:52:387 EDT] 00000027 ReplicatedPar I   CWOBJ1511I:
IBM_SYSTEM_xsastats.server:stats:0 (primary) is open for business.
[11/2/10 23:39:52:387 EDT] 00000028 ReplicatedPar I   CWOBJ1511I:
session:IBM_SYSTEM_ENTITYMANAGER_MAPSET:0 (primary) is open for business.
[11/2/10 23:39:54:559 EDT] 00000029 ReplicatedPar I   CWOBJ1511I:
session:sessionMapSet:0 (primary) is open for business.
[11/2/10 23:39:54:762 EDT] 0000002a ReplicatedPar I   CWOBJ1511I:
session:sessionMapSet:1 (primary) is open for business.
[11/2/10 23:39:54:809 EDT] 0000002b ReplicatedPar I   CWOBJ1511I:
session:sessionMapSet:2 (primary) is open for business.
[11/2/10 23:39:55:293 EDT] 0000002c ReplicatedPar I   CWOBJ1511I:
session:sessionMapSet:3 (primary) is open for business.
[11/2/10 23:39:55:699 EDT] 0000002d ReplicatedPar I   CWOBJ1511I:
session:sessionMapSet:4 (primary) is open for business.
```

5.3.3 Configuring WebSphere Portal Server

With the eXtreme Scale infrastructure ready, configure the Portal Server to offload session management to the eXtreme Scale grid, as shown in Figure 5-10 on page 86.

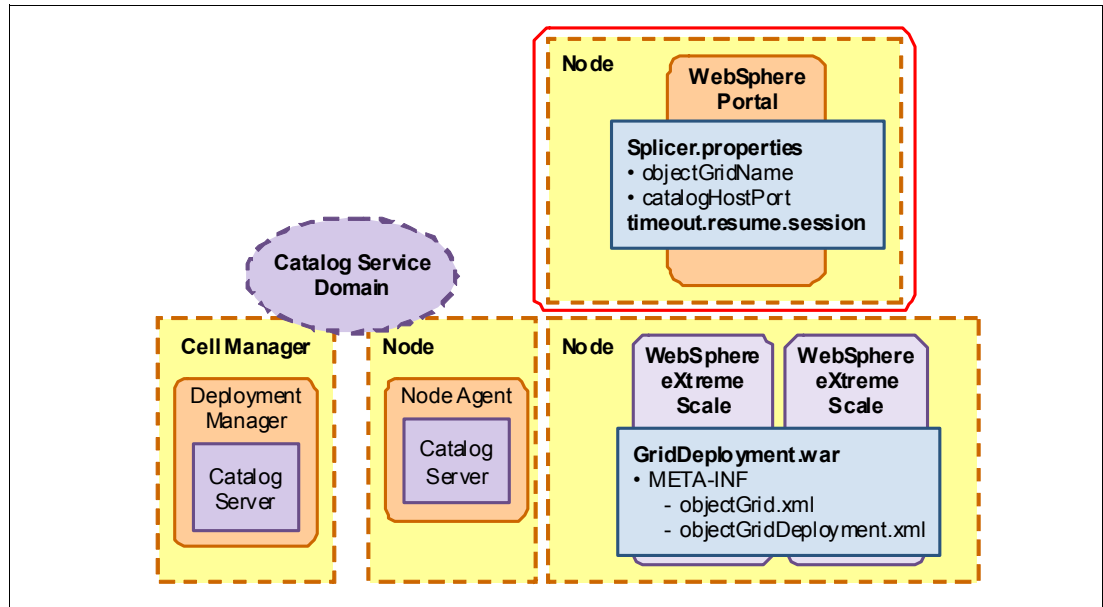


Figure 5-10 Configuring Portal Server with eXtreme Scale

To configure the Portal application servers to use the eXtreme Scale grid for session management, you must pass the session manager configuration parameters for your application to the session manager. These parameters can be in the form of servlet context initialization parameters in the deployment descriptors. However, when running in the WebSphere Portal Server environment, you do not need to do this manually for your applications. Instead, you can use the `splicer.properties` file to hold the configuration and define the file to the application servers with custom properties. The custom properties include:

- ▶ The `wps.com.ibm.websphere.xs.sessionFilterProps` property, which is set to the location of the `splicer.properties` file.
- ▶ For each custom portlet, define the `Application_Name.com.ibm.websphere.xs.sessionFilterProps` property and set it to the location of `splicer.properties` file.

Important: The administrative portlets that come with WebSphere Portal Server are not all serializable, so they cannot be persisted and should not be included in the applications that get spliced. If you specify the property `com.ibm.websphere.xs.sessionFilterProps` (without the application name) and set its value to the location of the `splicer.properties` file, Portal will try to splice all portlets including the administrative portlets, which can impact the way your environment performs.

The steps to configure the `splicer.properties` file and configure the application server to use them are as follows:

1. The `splicer.properties` file contains all the configuration options for a servlet-filter-based grid session manager. Follow these steps to add this file to WebSphere Portal Server:
 - a. Locate the `splicer.properties` file in the following folder:
`WAS_HOME\optionalLibraries\ObjectGrid\session\samples`
 - b. Modify the values. Most values in this file will work using the default values. However, you will need to define the variables as shown in Example 5-9 on page 87.

Example 5-9 splicer.properties file for session management

```
# A string value of either "REMOTE" or "EMBEDDED". The default is REMOTE.
# If it is set to "REMOTE", the session data will be stored outside of
# the server on which the web application is running. If it is set to
# "EMBEDDED", an embedded WebSphere eXtreme Scale container will start
# in the application server process on which the web application is running.

objectGridType = REMOTE

# A string value that defines the name of the ObjectGrid
# instance used for a particular web application. The default name
# is session. This property must reflect the objectGridName in both
# the objectgrid.xml and deployment.xml files used to start the eXtreme
# Scale containers.

objectGridName = session

# Catalog Server can be contacted to obtain a client side
# ObjectGrid instance. The value needs to be of the
# form "host:port<,host:port>", where the host is the listener host
# on which the catalog server is running, and the port is the listener
# port for that catalog server process.
# This list can be arbitrarily long and is used for bootstrapping only.
# The first viable address will be used. It is optional inside WebSphere
# if the catalog.services.cluster property is configured.

catalogHostPort =
saw006-sys2.itso.ral.ibm.com:9809,saw006-sys2.itso.ral.ibm.com:2809
```

Review and change the following settings as appropriate.

- **objectGridType**
This variable can take a string value of EMBEDDED or REMOTE. It indicates whether the eXtreme scale grid container to be started is embedded inside the Portal Server process or is remote. In our sample environment the grid container was configured as REMOTE.
- **objectGridName**
This variable takes a string value used by Portal to identify which grid in the grid container is to be used. Note that this name should match the objectGridName specified in the objectGridStandAlone.xml and objectGridDeploymentStandAlone.xml files.
- **catalogHostPort**
Specifies the catalog server connection information. The value of this variable has to be of the form host:port,host:port, where the host is the listener on which the catalog server is running and port is the listener port for that catalog server process. This list can be any length and the first viable address is used.

Note: In our sample setup, because there are two instances of catalog servers, one on the deployment manager and one on the node agent, that we have to specify both in the `splicer.properties` file. The catalog servers listen on the `BOOTSTRAP_ADDRESS` of the server. By default, the `BOOTSTRAP_ADDRESS` of the deployment manager profile is 9809 and that of the node agent is 2809. All catalog servers need to be started at approximately the same time.

There are few more variables in `splicer.properties` that you might want to consider changing:

- `replicationInterval`

An integer that defines the time in seconds between updated sessions being written to the grid. 0 means updated sessions are written to the grid at the end of the servlet service method call for each request.

- `sessionTableSize`

An integer value that defines the number of session references kept in memory.

Important: The number of sessions kept in memory must be a value greater than the maximum web container thread pool setting (default 50). Otherwise the web container threads will be bottlenecked by the in-memory session cache. The default of the `sessionTableSize` setting is 2000, but it should be set to a number that makes sense for the specific deployment based on the application server heap settings and the average HTTP session size. The in-memory sessions are kept in an least recently used (LRU) cache, so after the number of sessions in the web container is higher than `sessionTableSize`, the least recently used session is invalidated from the web container, but the session data is still stored in the remote eXtreme Scale container. A subsequent request for this session will actually create a new HTTP session, and the previous session's data will be loaded into the newly created session.

- `fragmentedSession`

Takes a value of `true` or `false` to control whether session data is stored as a whole entry or each attribute separately.

- `securityEnabled`

Takes a value of `true` or `false` to enable or disable eXtreme Scale client security. This setting needs to match the `securityEnabled` setting in the eXtreme Scale server properties file. If the settings do not match, an exception occurs.

2. Next, add the `splicer.properties` file to your Portal application server.

- In the administrative console, click **Server** → **Server Types** → **WebSphere Application Servers** and select your Portal server.
- On the right side of the page, under Server Infrastructure click **Administration** → **Custom Properties**.
- Click **New**.
- Enter the following:
 - Name: `wps,com.ibm.websphere.xs.sessionFilterProps`
 - Value: Location of the `splicer.properties` file.
- Click **Apply** and go back to the Custom Properties page.

- f. Click **New**.
- g. Enter the following:
 - Name: *YourPortletName*.com.ibm.websphere.xs.sessionFilterProps
 - Value: Location of the splicer.properties file

Note: To know the exact name of your custom Portlet as recognized by Portal, on the administrative console, click **Server** → **Server Types** → **WebSphere application servers** → **WebSphere_Portal**. On the right side, under Applications, click **Installed applications**. This lists all of the applications installed on the Portal Server.

- h. Click **Apply** and save your changes.
- i. Repeat these steps for all the custom portlets. The Custom Properties page should look as shown in Figure 5-11.

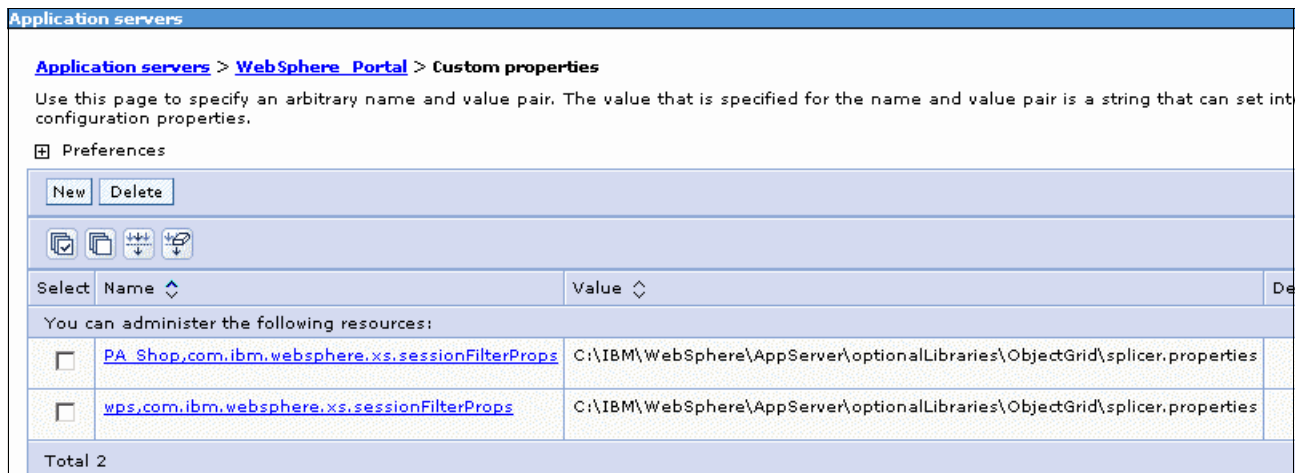


Figure 5-11 Custom properties for splicing custom portlets on the Portal application server

3. Add session timeout settings.

The `timeout.resume.session` parameter is not included in the `ConfigService.properties` file and by default the WebSphere Application Server administration equates this parameter to `false`. Therefore, if an idle session timeout is experienced, a user will see the `ErrorSessionTimeout` window and be forced to re-login. The login causes a new session to be created.

Set the `timeout.resume.session` property to `true`. By default, Portal will verify the session ID for an active user, and if the session ID changes, the application will require a re-login by the user. Because a user session can change when that session is invalidated from the session cache, this value needs to be `true` when using eXtreme Scale for session persistence.

To add the `timeout.resume.session` property to the Portal Server do the following:

- a. On the WebSphere administrative console, click **Resources** → **Resource Environment** → **Resource Environment Providers**.
- b. Open `WP ConfigService` and click **Custom Properties**.
- c. Click **New** and create a custom property with the name `timeout.resume.session` and the value `true`, as shown in the Figure 5-12 on page 90

Figure 5-12 Session timeout settings on Portal Server

- d. Click **Apply** and save your changes.
4. Restart the Portal Server.

5.3.4 Validating the eXtreme Scale integration

Now that the eXtreme Scale and Portal Integration environment is ready, validate the operational correctness of the environment. The best way to validate the operation is to check if session persistence is working with eXtreme Scale. We used the following procedure to conduct this test.

1. We created a simple portlet that adds data to the session, retrieves it from session, and displays it. Then we deployed and started the portlet.
2. From the messages in the `SystemOut.log` of the Portal server, the connection to the grid occurred as shown in Example 5-10.

Example 5-10 Log entries on the Portal application server showing the grid connection

```
[10/29/10 1:32:59:350 EDT] 00000079 HttpSessionFi A   CWWSM0007I: Using the
ObjectGrid based Session Manager.
[10/29/10 1:33:00:194 EDT] 00000079 JvmMemoryUtil I   CWOBJ4542I: Basic
BackingMap memory sizing is enabled.
[10/29/10 1:33:00:225 EDT] 00000079 ObjectGridImp I   CWOBJ4700I: The map name
objectgridSessionAttribute matched the regular expression of template map
```

objectgridSessionAttribute.*. The objectgridSessionAttribute map has been created for ObjectGrid session.

3. We accessed the portlet so that data was added to the session object. In our sample environment we added a few items to the cart, as shown in Figure 5-13.

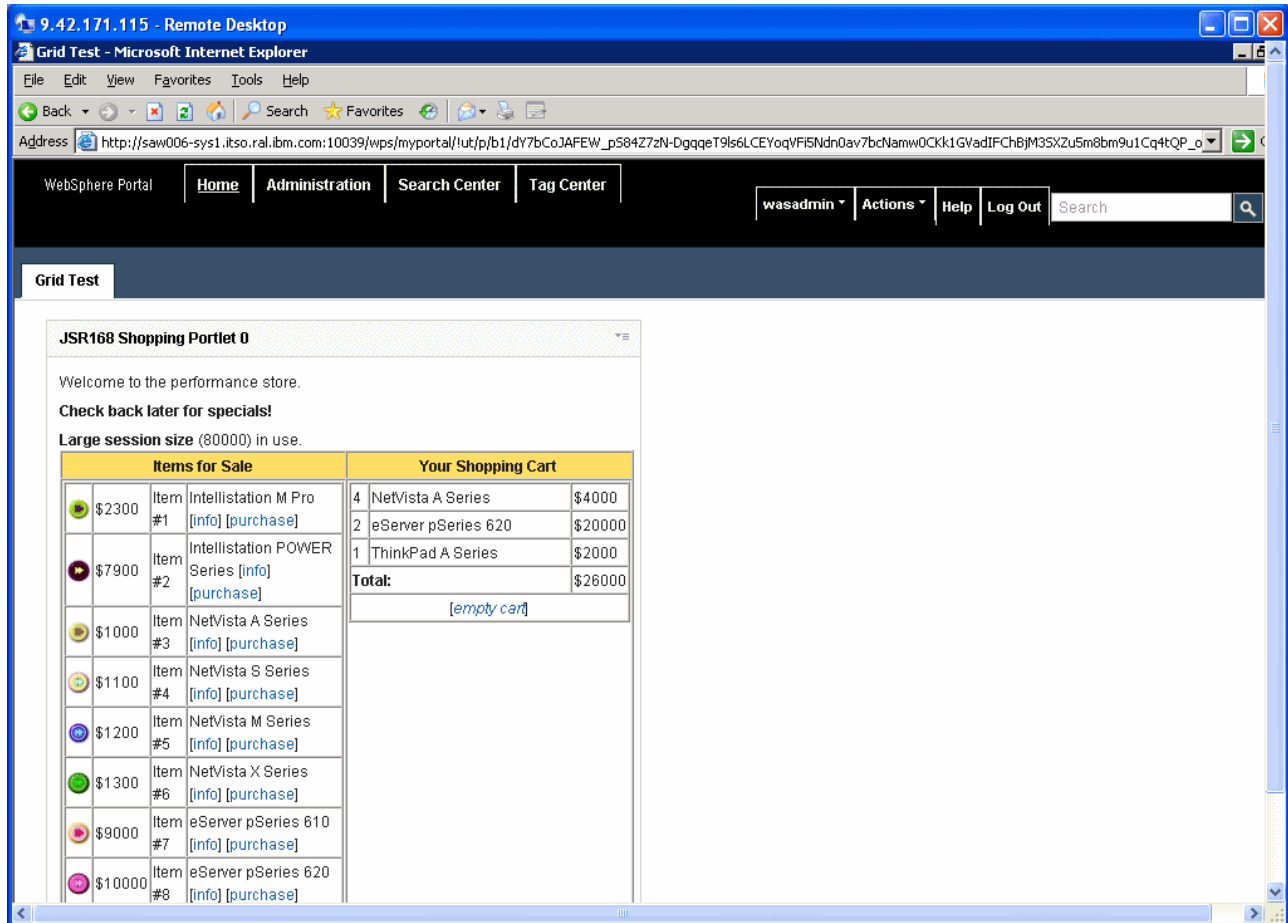


Figure 5-13 Sample Shopping Cart portlet

4. We viewed the **xsadmin** output to see if session data was stored in the grid. The **xsadmin** output from this sample environment is shown in Figure 5-14 on page 92.

This administrative utility is provided as a sample only and is not to be considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:9809

*****Displaying Results for Grid - session, MapSet - sessionMapSet*****

*** Listing Maps for saw006-sys2Cell101\saw006-sys2Node01\XS_GridCluster_Member1 ***

Map Name	Partition	Map Size	Used Bytes (KB)	Shard Type
objectgridSessionAttribute	5	21	12	AsynchronousReplica
objectgridSessionAttributeEvicted	5	4	80	AsynchronousReplica
objectgridSessionMetadata	0	0	0	Primary
objectgridSessionMetadata	1	0	0	Primary
objectgridSessionMetadata	2	0	0	Primary
objectgridSessionMetadata	3	0	0	Primary
objectgridSessionMetadata	4	0	0	Primary
objectgridSessionMetadata	5	4	1	AsynchronousReplica
objectgridSessionMetadata	6	0	0	AsynchronousReplica
objectgridSessionMetadata	7	0	0	AsynchronousReplica
objectgridSessionMetadata	8	0	0	AsynchronousReplica
objectgridSessionMetadata	9	0	0	AsynchronousReplica

Server Total: 29 <95KB>

*** Listing Maps for saw006-sys2Cell101\saw006-sys2Node01\XS_GridCluster_Member2 ***

Map Name	Partition	Map Size	Used Bytes (KB)	Shard Type
objectgridSessionAttribute	5	21	12	Primary
objectgridSessionAttributeEvicted	5	4	80	Primary
objectgridSessionMetadata	0	0	0	AsynchronousReplica
objectgridSessionMetadata	1	0	0	AsynchronousReplica
objectgridSessionMetadata	2	0	0	AsynchronousReplica
objectgridSessionMetadata	3	0	0	AsynchronousReplica
objectgridSessionMetadata	4	0	0	AsynchronousReplica
objectgridSessionMetadata	5	4	1	Primary
objectgridSessionMetadata	6	0	0	Primary
objectgridSessionMetadata	7	0	0	Primary
objectgridSessionMetadata	8	0	0	Primary
objectgridSessionMetadata	9	0	0	Primary

Server Total: 29 <94KB>

Total Domain Count: 58 <190KB>

Figure 5-14 Sample xsadmin output

For details on how to use **xsadmin**, see Section 5.4.1, “Monitoring eXtreme Scale with the xsadmin utility” on page 93.

5. In a failover situation, users need be able to retrieve the data they added to the session before the failover. We restarted the Portal application server to initiate a failover scenario.
6. Then we refreshed the Shopping Cart portlet and checked to see if the items we added to the cart before restarting the server had persisted.

5.4 Monitoring

There are several options and interfaces available for monitoring an eXtreme Scale deployment environment:

- ▶ The Statistics API, a direct interface to eXtreme Scale's internal statistics tree (see <http://publib.boulder.ibm.com/infocenter/wxinfo/v7r1/index.jsp?topic=/com.ibm.websphere.e.extremescale.prog.doc/txsmonitorstats.html>)
- ▶ The WebSphere Application Server performance monitoring infrastructure (PMI) and Tivoli Performance Viewer
- ▶ MBeans statistics
- ▶ The **xsadmin** utility
- ▶ Logs and traces
- ▶ Tools like IBM Tivoli Monitoring, Hyper HQ and CA Wily Introscope

Most of these statistics monitoring options are built on an internal statistics tree but are used for different reasons. In this chapter we will be focusing on the **xsadmin** utility and PMI to monitor session management on eXtreme Scale. For more information about other monitoring options, see Section 3.8, “Monitoring eXtreme Scale” on page 47.

5.4.1 Monitoring eXtreme Scale with the xsadmin utility

WebSphere eXtreme Scale version 7.0 includes a sample **xsadmin** administration utility. Using this utility, you can monitor the internal state of your eXtreme Scale grid, view diagnostic information, and stop container servers. It also provides a method for parsing and discovering current deployment data, and can be used as a foundation for writing custom utilities.

In this section, we will only cover the basic **xsadmin** command line options to monitor session data in the grid. Further information about **xsadmin** can be found in the Information Center at:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/txsxsadmin.html>

To view the HTTP session statistics in the grid, you need to place data in the grid first. Because we have offloaded session data for our custom portlet to eXtreme Scale, any sessions created in Portal for that portlet will be stored in the grid. To illustrate how we monitor with **xsadmin**, we log into Portal Server and load the page containing the custom portlet so that a session is placed in the grid.

1. Locate the **xsadmin** utility in the *WAS_HOME\bin* directory.
2. To display a list of all the grid containers run the following command:

```
xsadmin.bat -dmgr -containers
```

Depending on your grid deployment configuration, you will see a list of containers similar to Figure 5-15 on page 94.

```

This administrative utility is provided as a sample only and is not to be
considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:9809

*** Show all online containers for grid - session & mapset - sessionMapSet
Host: saw006-sys2.itso.ral.ibm.com
Container: saw006-sys2Cell101\saw006-sys2Node01\XS_GridCluster_Member1_C-24, Se
rver:saw006-sys2Cell101\saw006-sys2Node01\XS_GridCluster_Member1, Zone:DefaultZon
e
  Partition Shard Type
  5          AsynchronousReplica
  6          AsynchronousReplica
  7          AsynchronousReplica
  8          AsynchronousReplica
  9          AsynchronousReplica
  0          Primary
  1          Primary
  2          Primary
  3          Primary
  4          Primary
Container: saw006-sys2Cell101\saw006-sys2Node01\XS_GridCluster_Member2_C-19, Se
rver:saw006-sys2Cell101\saw006-sys2Node01\XS_GridCluster_Member2, Zone:DefaultZon
e
  Partition Shard Type
  0          AsynchronousReplica
  1          AsynchronousReplica
  2          AsynchronousReplica
  3          AsynchronousReplica
  4          AsynchronousReplica
  5          Primary
  6          Primary
  7          Primary
  8          Primary
  9          Primary

Num containers matching = 2
Total known containers = 2
Total known hosts      = 1

```

Figure 5-15 List of containers shown by Xsadmin

Tip: The **-dmgr** option is required when connecting to a catalog server hosted by any WebSphere Application Server process or cluster of processes. Because we have the catalog server hosted in a WebSphere Application Server process for our sample environment, we will use the **-dmgr** option for all executions of the **xsadmin** command.

3. To see the number of entries in all the maps with mapsizes for the grid, run the following command:

```
xsadmin.bat -dmgr -g session -mapsizes
```

Note that **session** is the objectGridName as specified in the objectGrid.xml and objectGridDeployment.xml files.

A sample output for this command will look like Figure 5-16 on page 95.

This administrative utility is provided as a sample only and is not to be considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:9809

*****Displaying Results for Grid - session. MapSet - sessionMapSet*****

*** Listing Maps for saw006-sys2Cell101\saw006-sys2Node01\XS_GridCluster_Member1 ***

Map Name	Partition	Map Size	Used Bytes (KB)	Shard Type
objectgridSessionAttribute	5	21	12	AsynchronousReplica
objectgridSessionAttributeEvicted	5	4	80	AsynchronousReplica
objectgridSessionMetadata	0	0	0	Primary
objectgridSessionMetadata	1	0	0	Primary
objectgridSessionMetadata	2	0	0	Primary
objectgridSessionMetadata	3	0	0	Primary
objectgridSessionMetadata	4	0	0	Primary
objectgridSessionMetadata	5	4	1	AsynchronousReplica
objectgridSessionMetadata	6	0	0	AsynchronousReplica
objectgridSessionMetadata	7	0	0	AsynchronousReplica
objectgridSessionMetadata	8	0	0	AsynchronousReplica
objectgridSessionMetadata	9	0	0	AsynchronousReplica

Server Total: 29 <95KB>

*** Listing Maps for saw006-sys2Cell101\saw006-sys2Node01\XS_GridCluster_Member2 ***

Map Name	Partition	Map Size	Used Bytes (KB)	Shard Type
objectgridSessionAttribute	5	21	12	Primary
objectgridSessionAttributeEvicted	5	4	80	Primary
objectgridSessionMetadata	0	0	0	AsynchronousReplica
objectgridSessionMetadata	1	0	0	AsynchronousReplica
objectgridSessionMetadata	2	0	0	AsynchronousReplica
objectgridSessionMetadata	3	0	0	AsynchronousReplica
objectgridSessionMetadata	4	0	0	AsynchronousReplica
objectgridSessionMetadata	5	4	1	Primary
objectgridSessionMetadata	6	0	0	Primary
objectgridSessionMetadata	7	0	0	Primary
objectgridSessionMetadata	8	0	0	Primary
objectgridSessionMetadata	9	0	0	Primary

Server Total: 29 <94KB>

Total Domain Count: 58 <190KB>

Figure 5-16 Map entries for a grid shown by xsadmin

5.4.2 Monitoring eXtreme Scale with the Tivoli Performance Viewerq

WebSphere eXtreme Scale supports Performance Monitoring Infrastructure (PMI) when running on WebSphere Application Server. PMI collects performance data on runtime applications and provides interfaces that support external applications to monitor performance data. You can use the Tivoli Performance Viewer in the WebSphere administrative console to view the collected data. For information about using PMI to monitor the grid, see 3.8.3, “Monitoring with the Tivoli Performance Viewer” on page 50.

You can use PMI to monitor your environment only when you are using WebSphere eXtreme Scale with WebSphere Application Server. If you have a stand-alone deployment of eXtreme Scale then you need to use the eXtreme Scale web console (see Section 3.8.1, “Using the eXtreme Scale web console” on page 47).

5.5 Guidelines and best practices

In this section, we discuss a few guidelines in terms of coding, deployment, and tuning to get the maximum benefit from your eXtreme Scale session grid.

5.5.1 Deployment

Planning the capacity of an eXtreme Scale environment is a crucial task. If an initial and a projected data set size is known, you can plan for the right capacity for an eXtreme Scale deployment to maximize the elasticity and achieve maximum benefits.

In this section, we talk about how to calculate the required servers, recommended partitions, shards, and number of JVMs for an eXtreme Scale deployment of a session grid based on certain application inputs.

We will need the following to begin with:

- ▶ iSize: The initial grid size represented in number of objects expected to be stored in the grid.
- ▶ gRate: Annual growth rate.
- ▶ rFactor: The number of replicas required in the deployment.
- ▶ sizeKb: The average object size in Kb.
- ▶ jHeap: Java heap size in Gb.
- ▶ nShards: Number of shards to be placed in each JVM.
- ▶ phMem: Physical machine memory in Gb.

We are also assuming the following constant values:

- ▶ mHc: Maximum Java heap consumption, a constant value of 70%.
- ▶ hRm: Additional Java heap real memory requirement, a constant value of 25%.
- ▶ oSMo: Operating system memory overhead in Gb, a constant of 1.5 Gb.

Given these inputs and constants we can arrive at the following recommended values:

- ▶ Raw Memory required in Kb (rMem):
$$rMem = iSize * sizeKb * (rFactor + 1)$$
- ▶ Recommended number of JVMs (rJVM):
$$rJVM = \text{RoundUp}(rMem / ((1000000 * jHeap) * mHc))$$
- ▶ Recommended number of Shards (rShards):
$$rShards = rJVM * nShards$$
- ▶ Recommended Partitions (rPar):
$$rPar = \text{GetNearestPrime}(rShards / (1 + rFactor))$$
- ▶ Minimum required Servers (rServ):
$$rServ = \text{RoundUp}(rJVM / ((phMem - oSMo) / (jHeap * (1 + hRm))))$$

These values help in designing the deployment of eXtreme Scale for session offloading with Portal Server.

5.5.2 Application code

When coding applications that you plan to use with eXtreme Scale, remember the following guidelines:

- ▶ All session objects that need to be stored in the grid have to be serializable, that is, they must implement the `java.io.Serializable` interface.
- ▶ Set all application objects or data that needs to be recovered in the event of a failover on the HTTP session object. A common mistake that developers make is to forget to save an object to session after an update.

Example 5-11 on page 97 shows code that retrieves a `shoppingCart` `ArrayList` attribute from the session object and updates it. After the update, the shopping cart list is never updated into

the session. If a failover occurs in this case, the session gets invalidated out of the session cache and gets reconstituted from the grid under a new session, and the shoppingCart data is lost.

Example 5-11 Bad Code: storing application data in session

```
ArrayList list = httpSession.getAttribute("shoppingCart");  
list.add("item1");
```

Example 5-12 shows the correct way of doing this. After the ArrayList has been updated with an added item, it is placed back in the session object. In essence, any update to an object that is stored in the session should be directly updated in the session so it can be pushed out to the grid.

Example 5-12 Good Code: storing application data in session

```
ArrayList list = httpSession.getAttribute("shoppingCart");  
list.add("item1");  
httpSession.setAttribute(list);
```

5.5.3 JVM tuning

Java virtual machine (JVM) tuning can yield a significant improvement in your deployment of WebSphere eXtreme Scale. If you have a large amount of objects that are being stored in WebSphere eXtreme Scale, adjust the heap size to an appropriate level to avoid running out of memory.

Garbage collection

WebSphere eXtreme Scale creates temporary objects that are associated with each transaction, such as request and response, and log sequence. Because these objects affect garbage collection efficiency, tuning garbage collection is critical. For the IBM virtual machine for Java, use the optavgpause collector for high update rate scenarios (100% of transactions modify entries). The gencon collector works much better than the optavgpause collector for scenarios where data is updated relatively infrequently (10% of the time or less). You will need to experiment with both collectors to see what works best for your application.

WebSphere eXtreme Scale can run on various versions of Java 2 Platform, Standard Edition (J2SE). WebSphere eXtreme Scale Version 6.1 supports J2SE Version 1.4.2 and later. For improved developer productivity and performance, use J2SE 5 or later to take advantage of annotations and improved garbage collection. WebSphere eXtreme Scale works on 32-bit or 64-bit Java virtual machines.

Large heaps

When the application needs to manage a large amount of data for each partition, then garbage collection might be a factor. A mostly “read” scenario performs well even with large heaps (20 GB or more) if a generational collector is used. However, after the tenure heap fills, a pause proportional to the live heap size and the number of processors on the box occurs. This pause can be large on smaller boxes with large heaps.

Tip: WebSphere eXtreme Scale also supports WebSphere Real Time Java. With WebSphere Real Time Java, the transaction processing response for WebSphere eXtreme Scale is more consistent and predictable, and the impact of garbage collection and thread scheduling is minimized. The impact is reduced to the degree that the standard deviation of response time is less than 10% of regular Java. For more information about WebSphere Real Time, go to the following address:

<http://www-01.ibm.com/software/webservers/realtime>



Integrating WebSphere Commerce with WebSphere eXtreme Scale

WebSphere eXtreme Scale provides business and technical benefits to new and existing WebSphere Commerce deployments. This chapter provides an overview of those benefits and how to configure WebSphere Commerce to use WebSphere eXtreme Scale. It also covers the technical aspects of installation, configuration, and monitoring.

This chapter includes the following sections:

- ▶ 6.1, “Why use eXtreme Scale with WebSphere Commerce?” on page 100
- ▶ 6.2, “Overview of caching differences with eXtreme Scale” on page 100
- ▶ 6.3, “How to integrate eXtreme Scale with WebSphere Commerce” on page 107
- ▶ 6.4, “Monitoring considerations” on page 132

6.1 Why use eXtreme Scale with WebSphere Commerce?

If you already have an existing WebSphere Commerce infrastructure, you might be wondering why you would introduce another product into the mix simply to provide the same user functionality. This section will discuss how adding WebSphere eXtreme Scale to WebSphere Commerce sites can bring a range of technical and business benefits.

WebSphere eXtreme Scale can be used in a wide range of scenarios. A number of these apply to WebSphere Commerce. For example, the WebSphere eXtreme Scale API can be used as a side cache or to cache a search engine accessed through a web service. However, that use case entails application development. The good news is that you can benefit from WebSphere eXtreme Scale without any code change whatsoever and with limited impact to an existing Commerce environment.

WebSphere Application Server (versions 7.0.0.5 and later, and v6.1.0.27 and later) provides support for the offload of dynamic cache content from the default implementation to WebSphere eXtreme Scale. Any WebSphere application can benefit from this. Typically, applications with expensive content to generate, such as large web pages, large web service responses, or expensive computational logic, benefit the most from the dynamic cache functionality. The offload to WebSphere eXtreme Scale becomes more and more useful as you need to handle large amounts of cached data or when trying to scale your environment to more users.

WebSphere Commerce makes heavy use of the dynamic cache functionality to optimize the performance of rendering web and catalog content. As of WebSphere Commerce Server v7.0.0.1, the eXtreme Scale integration has been made available to WebSphere Commerce sites and stores.

The integration between WebSphere Commerce (specifically the dynamic cache feature) and WebSphere eXtreme Scale is straightforward. WebSphere eXtreme Scale is another dynamic cache provider. To use eXtreme Scale, you simply need to change a few parameters on the Commerce application server to change the dynamic cache provider and therefore route cached content to eXtreme Scale instead. That means that you do not have to change the Commerce application at all and can continue to use existing management tooling and anything that has relied on the dynamic cache API, such as edge caching.

6.2 Overview of caching differences with eXtreme Scale

This section gives an overview of how caching with WebSphere eXtreme Scale differs from using the default dynamic cache provider. We provide a brief introduction to the dynamic cache provider, but will not cover this in detail here as this is done well in other publications, particularly in *Mastering DynaCache in WebSphere Commerce*, SG24-7393.

6.2.1 Dynamic cache service

We are familiar with the benefits of caching data. The biggest performance gains for an application are found when data can be held ready for use or even where business logic or page rendering simply does not need to be done.

The dynamic cache service (also known as *dynacache*) within the WebSphere Application Server is a powerful data cache and is best known for storing dynamically generated web pages. The reason this is so powerful is that you can benefit from the dynamic cache provider without having to change your web application. You can simply tell WebSphere Application

Server which pages can be cached, under what conditions, and for how long it is valid to cache it. An overview of the dynamic cache service is shown in Figure 6-1.

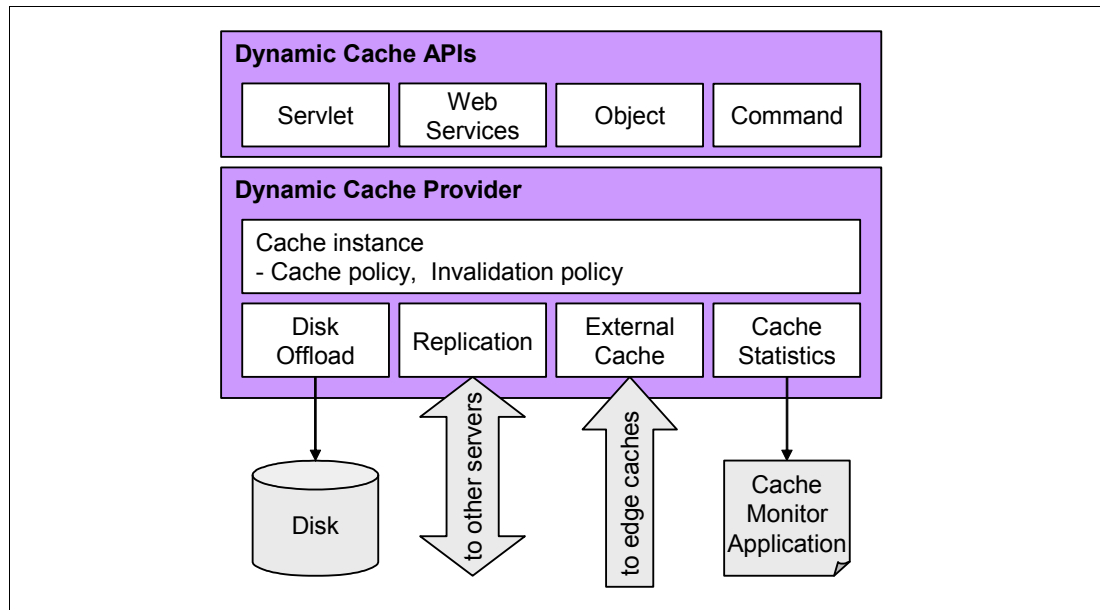


Figure 6-1 Technical overview of the dynamic cache service

The dynamic cache service is not restricted to just web content. The dynamic cache service has a number of APIs that can be used in a variety of ways for separate parts of an enterprise Java application. Let's take a look at the options, which are illustrated in Figure 6-1:

- Web content

We can provide an XML file, `cachespec.xml`, in a web module to list a range of dynamic web pages and web fragments that can be cached. This includes all types of dynamically generated web content such as servlets, JSPs, JSFs, portlets, and AJAXs. This option does not require code change to use dynamic caching.

- Web services

Web services are naturally also web content, but it is worth mentioning them separately as there is specific support for being able to cache web service responses based on varying data in SOAP headers and parameters. This option does not require code change to use dynamic caching.

- Java objects

WebSphere Application Server provides an API for storing Java objects called the DistributedMap API. An application can use the DistributedMap to cache data to reduce expensive data accesses from data sources such as web services or databases.

- Java commands

WebSphere Application Server provides the Command cache API to cache the output of business logic methods. When the API has been implemented, the cache can be configured and tuned with the `cachespec.xml` file in the same way as the web content.

All of the dynamic cache service APIs can benefit from the services that the dynamic cache provider implements (the services are shown in Figure 6-1):

- Cache replication copies data between application servers. It is configurable as to how much data is replicated, from cache invalidations to replicating all cache content.

- Cache invalidation management is available through automatic time-based, memory-based invalidation, programmatic or manual invalidation through the cache monitor, and specifying URLs that can invalidate a cache entry.
- Disk offload for when you need to cache more data than can be contained in the application server Java virtual machine.

The dynamic cache service can be used by any WebSphere Application Server application.

WebSphere Commerce makes good use of the dynamic cache service to optimize the web store browsing experience and for storing user-related data in object caches. The dynamically generated content is stored in an instance of the dynamic cache called a servlet cache. The Java objects are stored in instances of the dynamic cache called an object cache. These cache instances can be created and managed under the Resources section of the administrative console. By default, all cached data is stored in the cache instance created by default, called `baseCache`.

6.2.2 What to expect when using WebSphere eXtreme Scale

The integration between WebSphere Commerce and WebSphere eXtreme Scale does not require any change to applications using the dynamic cache API. But there are small changes to the behavior and the way the cache is managed. The differences in the architecture are illustrated in Figure 6-2 on page 103 and Figure 6-3 on page 103 and provide the background for comparing the difference.

Figure 6-2 on page 103 shows a typical deployment of the default dynamic cache provider. You can see that each application server contains the same cached data. This is replicated between the application servers on a best-efforts basis. In reality, the dynamic cache service provides a variety of replication options. First, it can copy all data to every server, so all servers contain the same data. But that is expensive to do. Alternatively, it can just publish invalidation events to every server, so that servers will not retain stale data. This is more efficient replication but means each server has to create their own cache entry. Regardless of the sharing mode, as the site reaches a steady state, each Commerce server will contain the same cached data. Also, every server will only contain a subset of the whole cache, with every server maintaining a disk off load of the cache. This is slower than an in-memory cache, but quicker than having to recreate the cache entry every time. However, for a large environment, that can be a significant disk requirement.

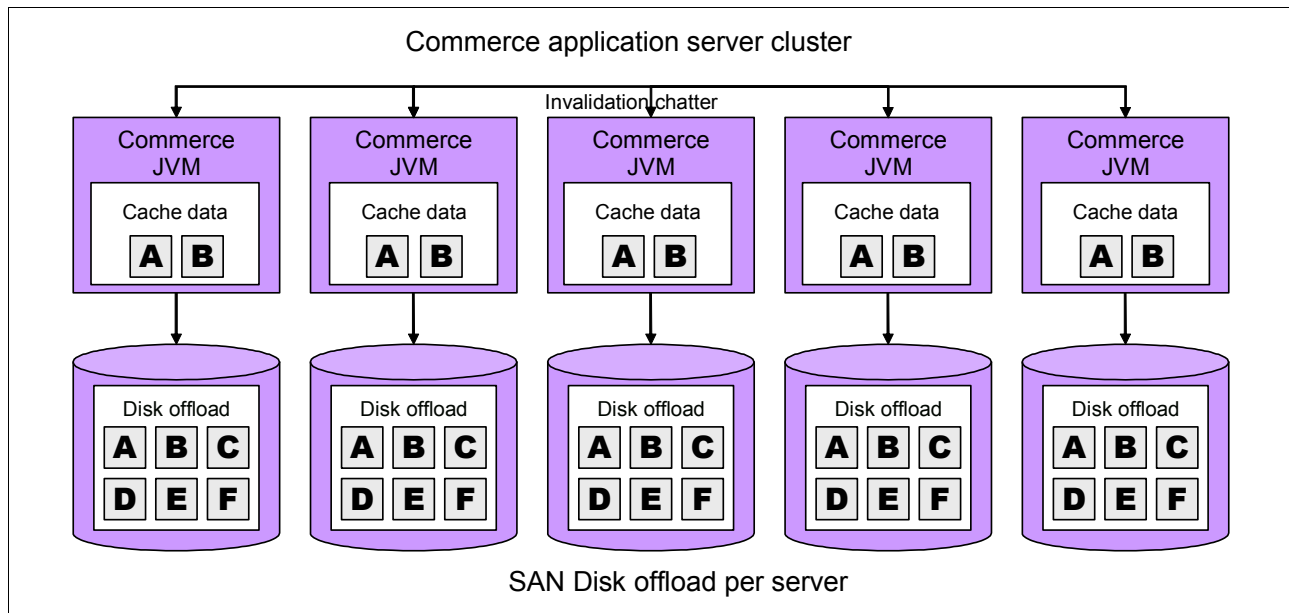


Figure 6-2 Example dynamic cache environment

Figure 6-3 shows a remote topology deployment of the WebSphere eXtreme Scale dynamic cache provider. You can see that all data is placed in WebSphere eXtreme Scale and that the Commerce application servers have no responsibility for cache management. There is no cache replication traffic or disk off load. In contrast, all cache accesses are over the network. Therefore, provide good, reliable bandwidth for access to eXtreme Scale. The overhead of remote access is mitigated by compressing the content in the grid (around 2.5x or 3x) so that it is served efficiently. With WebSphere Commerce, portions of the cache (the data caches) are assumed to be local to the application and are therefore create a large amount of traffic. We will also demonstrate how to tune this so that it is not a problem.

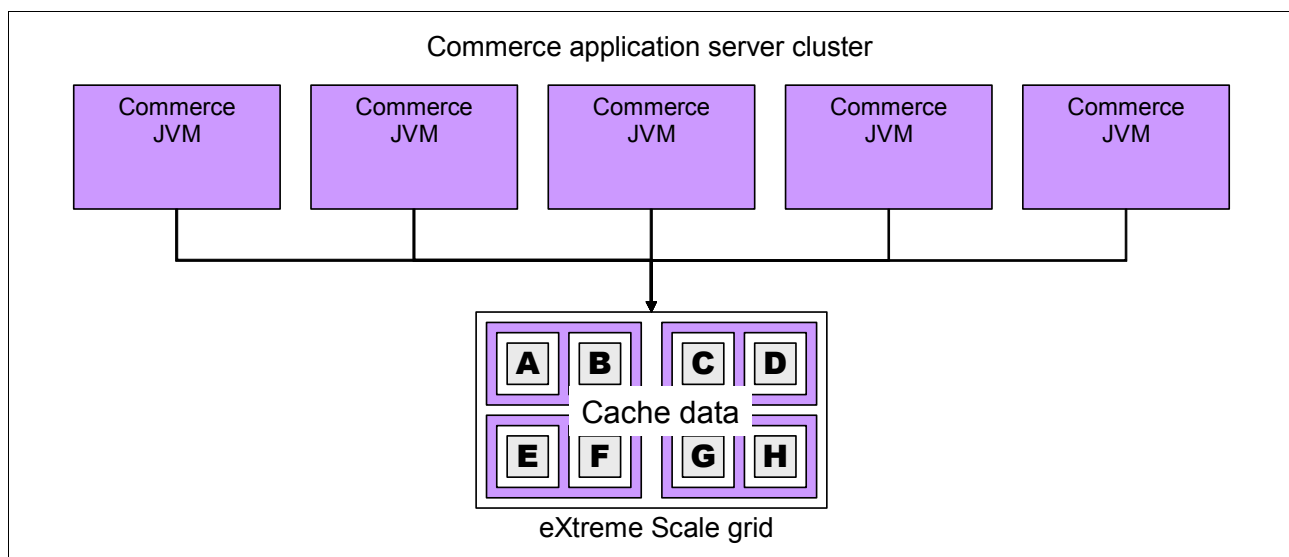


Figure 6-3 Example eXtreme Scale environment

We are going to highlight the main technical differences that will impact the design, configuration and management of a WebSphere Commerce deployment. For a more complete summary of the technical differences between the default dynamic cache provider

and WebSphere eXtreme Scale, see the WebSphere eXtreme Scale Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.webSphere.extremescale.over.doc/cxsdynacache.html>

The architectural diagrams (Figure 6-2 on page 103 and Figure 6-3 on page 103) help illustrate the benefits of moving to eXtreme Scale. “Architectural differences” on page 105 highlights the main differences to consider specifically when moving the dynamic cache provider to eXtreme Scale.

Topologies

There are four topologies available for the WebSphere eXtreme Scale dynamic cache provider.

► Local

This is the default behavior achieved when defining eXtreme Scale as the dynamic cache provider and is the topology that will always be used if cache replication is not enabled (this configuration is covered in “Configure the WebSphere Commerce application servers” on page 122). This option replaces the default dynamic cache service cache provided in the WebSphere Application Server with a local grid cache, with similar features and functions. It offers better performance than the default dynamic cache provider on large multi-processor enterprise servers because the eXtreme Scale code path is designed to maximize in-memory cache concurrency.

► Embedded

The embedded topology is similar to the default dynamic cache when distributed across a cluster. Embedded cache instances keep a full copy of the cache within each Commerce process that accesses the dynamic cache service, allowing all read operations to occur locally. All write operations go to a single server in which the transactional locks are managed before being replicated to the rest of the servers. This topology is better for workloads where cache-read operations greatly outnumber cache-write operations. With the embedded topology, you are limited to the amount of cache you can store in a Commerce JVM.

► Embedded_Partitioned

The embedded partitioned topology keeps all of the cache data within the Commerce processes. However, each process only stores a portion of the cache data. Most requests to the cache will be accessed remotely, resulting in a higher latency for read operations than the embedded topology. With this topology, the maximum size of the cache is not bound by the size of a single WebSphere process. Rather, it is the aggregate size of all the processes, minus the overhead of running the Commerce application. The embedded partitioned or remote topologies should be used when cache size requirement is greater than a single JVM and for workloads where cache-writes occur as often as, or more frequently than, reads.

► Remote

With the remote topology, the cache data is stored outside of the Commerce JVMs, providing a deployment model where the amount of memory available for caching is unrestricted. Commerce JVMs do not perform the two distinct workloads of application serving and caching, improving the efficiency of the Commerce application and the caching infrastructure.

Leading practice for Commerce: For WebSphere Commerce, a remote topology as illustrated in Figure 6-3 on page 103 is usually the best choice. Commerce deployments typically require large (greater than 1 JVM) caches, so the embedded_partitioned and remote topologies are needed for scaling the cache size. Out of those two topologies, the remote topology offers the greatest flexibility to scale the Commerce and caching tiers independently and, as a result of moving the cache out of the Commerce JVMs, make the Commerce application servers more efficient. The other tangible benefit of using the remote topology is that the eXtreme Scale licensing is only required on the remote grid, which makes it a cost-efficient approach.

Architectural differences

You will see the following differences in architecture when moving to dynamic cache and eXtreme Scale:

- ▶ No change to application

The application continues to use the dynamic cache provider and does not need to be changed.

- ▶ Remote cache

With the remote topology, all cache access is remote. There is currently no in-memory cache, or “near cache” for the dynamic cache plug-in. As a result, there will be a heavier requirement of network bandwidth to the cache than previously.

For more advanced topologies, it is possible to host a number of grids in eXtreme Scale. These can be for other integration scenarios such as the Portal integration scenario, applications with custom coding, or even multiple dynamic cache grids.

- ▶ eXtreme Scale is partitioned

Because the eXtreme Scale grid is partitioned to allow it to scale, there are factors in sizing (6.5, “Sizing guidance” on page 137), and monitoring (6.4, “Monitoring considerations” on page 132) that need to be taken into consideration.

- ▶ Distributed cache instead of cache replication

The default dynamic cache provider is a replicated cache instead of a distributed cache. Therefore certain of the modes that the dynamic cache provider allowed were to address performance impacts of replicating lots of data around an environment. Often environments needed to be configured to just replicate cache invalidations, or just replicate data when it is needed by another server (cache replication modes “not shared”, “pull/push” respectively).

By contrast, eXtreme Scale in a remote topology only has a single primary copy of the data, which the applications *all* have access to. So there will not be the scenario where more than one server has to create the cache entry with the potential issues that introduces. The impact of this is just a different (and often simpler) approach to configuring caching.

- ▶ Cache availability

Instead of replicating an entire cache throughout a Commerce environment, eXtreme Scale can be configured to maintain a copy of each partition on a separate server. It is configurable as to whether this is synchronous or asynchronous and how many copies (replicas) are needed. One asynchronous replica will give sufficient availability in most situations.

Performance and throughput

As with all performance and throughput comparisons, each environment is unique and will produce varying results. It is sufficient to underline at this point that the removal of the cache from Commerce into eXtreme Scale will allow the Commerce JVMs to run more efficiently and support greater throughput. A single request might not be significantly faster, but, with eXtreme Scale, usually there will be improved consistency of performance due to dramatically improved cache and environment stability.

More details on the performance and throughput are outlined in a study published on developerWorks that provides performance comparison metrics.

http://www.ibm.com/developerworks/websphere/techjournal/1008_genkin/1008_genkin.html

System availability

With WebSphere eXtreme Scale, we are adding two architectural components: the catalog server and the eXtreme Scale containers. We consider the availability of these components in the same way that we consider the availability of application servers in a cluster. Generally, configure the network as follows:

- ▶ Configure a minimum of three Catalog Server instances. These can run in the deployment manager and node agents and will add little overhead.
- ▶ Configure a minimum of three eXtreme Scale containers (application servers). This ensures that the data and requests will be spread evenly even after a grid ripple-start (a ripple start stops, then starts each application server in a cluster, one at a time).

Network availability, as in other areas of the infrastructure, is critical for operation of Commerce with eXtreme Scale.

Monitoring

The tools that are used today with Commerce, the Tivoli Performance Viewer and the cache monitor, continue to be appropriate with WebSphere eXtreme Scale. Certain statistics will need to be analyzed in a different context. For example, you might see aggregate values instead of per server values. For the cache monitor, use the Extended Cache Monitor from developerWorks because this works better with the eXtreme Scale functionality. More details can be found in 6.4, “Monitoring considerations” on page 132.

Restrictions

In principle you can continue to run your applications with the eXtreme Scale dynamic cache provider in exactly the same way as you do today. There are a few things that the eXtreme Scale dynamic cache provider will not do (such as receive dynamic cache events from the grid), but in a typical Commerce environment these will not affect the way you use the dynamic cache provider.

For full reference and discussion on these differences, see the section “Dynamic cache engine and eXtreme Scale functional differences” in the Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.over.doc/cxsdynacache.html>

Sizing

From a sizing perspective, an environment with eXtreme Scale is actually quite different. There is no disk cache and the data doesn’t need to exist in multiple places. Additionally, eXtreme Scale allows you to increase your cache size. You are able to cache greater volumes

of data such as pricing data that simply wasn't possible before. A simple sizing methodology is discussed in 6.5, "Sizing guidance" on page 137.

6.3 How to integrate eXtreme Scale with WebSphere Commerce

This section will give detailed steps on how to configure a typical WebSphere Commerce and WebSphere eXtreme Scale environment.

6.3.1 Sample environment

We have configured a sample topology to demonstrate the functionality provided by the eXtreme Scale dynamic cache provider. Figure 6-4 shows an architectural overview of the environment, demonstrating the key eXtreme Scale components overlaid on the WebSphere Commerce and application server components.

We are going to create a remote topology, so the eXtreme Scale grid will reside in separate application servers to the WebSphere Commerce application server. In the sample environment, the grid application servers were on the same node, although in reality these are best deployed on separate nodes for resilience. The WebSphere Commerce dynamic cache provider is set up to use this eXtreme Scale grid. Finally, we will make the catalog server resilient by running it in the deployment manager and node agents.

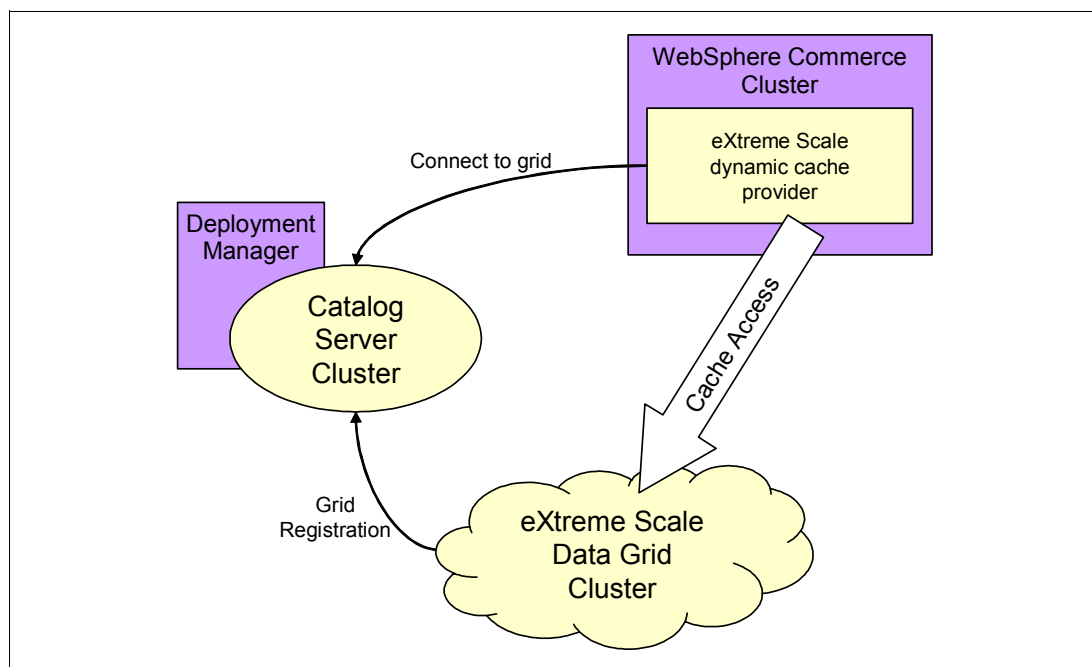


Figure 6-4 Architectural overview of the sample topology

6.3.2 Summary of configuration

Figure 6-5 on page 108 shows the three main areas that we need to adjust to configure the eXtreme Scale dynamic cache provider.

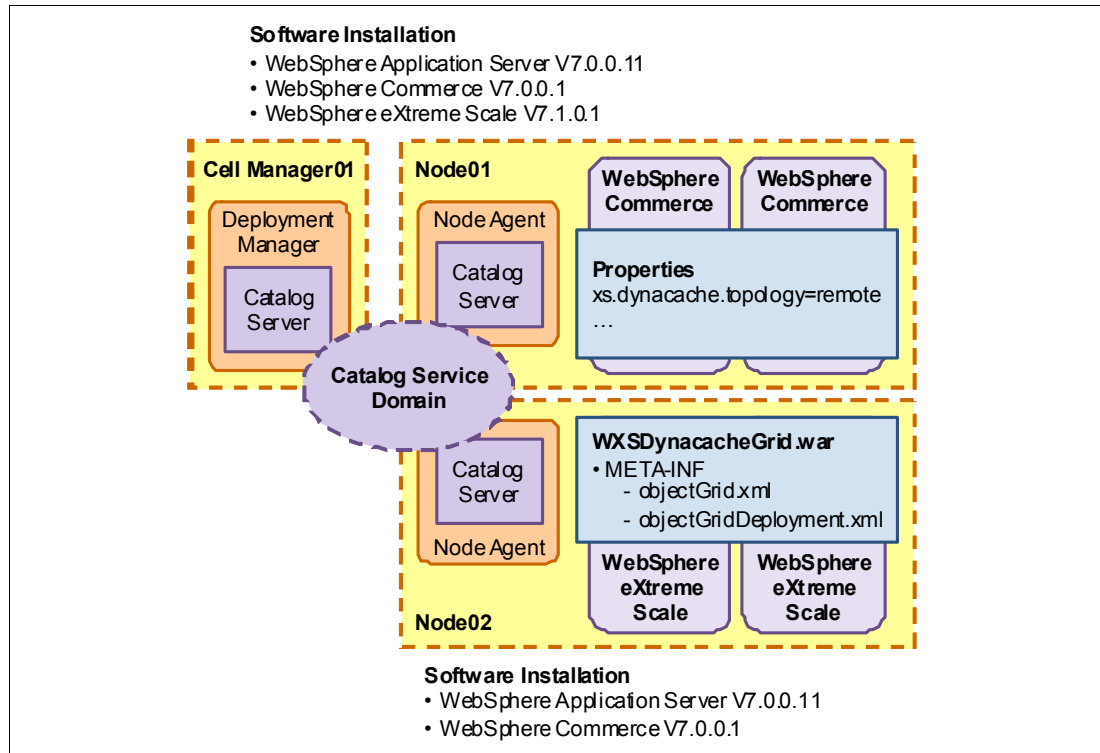


Figure 6-5 Operational view of the sample topology

The following are the six simple steps we used to build this integrated environment:

1. Install WebSphere eXtreme Scale on all nodes and create or augment profiles.
2. Configure and cluster the catalog service domain.
3. Create the eXtreme Scale dynamic cache cluster and deploy the WAR file containing the grid XML files.
4. Configure the Commerce application server(s) properties to use the eXtreme Scale grid.
5. Optimize performance by configuring Commerce data caches.
6. Start the environment in this order: catalog server, eXtreme Scale grid, Commerce cluster.

6.3.3 WebSphere eXtreme Scale prerequisites

The scenario is based on the following product releases:

- ▶ WebSphere Commerce v7.0.0.1
- ▶ WebSphere Application Server v7.0.0.11
- ▶ WebSphere eXtreme Scale V7.1.0.1 (V7.1 with cumulative fix 1)

The specific fix levels of products that are supported together will continually change, but at the time of writing, the versions that are certified to work together are as follows:

- ▶ WebSphere eXtreme Scale v7.1 (plus fixes PM21272, PM20613, included in cumulative fix 1)
- ▶ WebSphere eXtreme Scale v7.0 ifix 4 (plus fix PM21272)
- ▶ WebSphere Commerce 7.0.0.1
- ▶ WebSphere Commerce 6.0.0.7

- ▶ WebSphere Application Server v7.0.0.9 and later
- ▶ WebSphere Application Server v6.1.0.29 and later

In general, later fix levels will work together and are supported.

6.3.4 Installing WebSphere eXtreme Scale for integration with Commerce

In this section, we are only going to mention installation considerations specific to WebSphere Commerce and the use of the dynamic cache service. The WebSphere eXtreme Scale installation has been described in detail in Chapter 4, “Installing WebSphere eXtreme Scale” on page 57.

We are not going to cover the installation of WebSphere Commerce, but are going to make the following assumptions about the environment that we are going to install WebSphere eXtreme Scale into:

- ▶ The WebSphere Commerce installation has already been performed on WebSphere Application Server Network Deployment v7.0.0.11 (or any supported release).
- ▶ A WebSphere Commerce application server or cluster already exists that can be configured for use with eXtreme Scale.
- ▶ Although we are going to discuss how to set up the dynamic cache service on this cluster, we are not going to detail how to best configure the WebSphere Commerce site for caching. For more information, see *Mastering DynaCache in WebSphere Commerce*, SG24-7393.
- ▶ We are going to install WebSphere eXtreme Scale into the same Network Deployment cell as WebSphere Commerce. This is the simplest deployment, but introduces the operational consideration that eXtreme Scale and Commerce will have to continue to run on the same patch level of WebSphere Application Server.

At a high level, the following are the installation steps to prepare the Commerce environment:

1. Install WebSphere Application Server Network Deployment at the correct maintenance level to support WebSphere eXtreme Scale on the node(s) where you intend to create the WebSphere eXtreme Scale servers.
2. Install WebSphere eXtreme Scale on top of the WebSphere Commerce and the WebSphere Application Server installations. For specific version and patches, see 6.3.3, “WebSphere eXtreme Scale prerequisites” on page 108.

Installation option: There are two installation options for WebSphere eXtreme Scale; client and server. If you are installing a version of WebSphere eXtreme Scale earlier than version 7.1, you need to select the *server* option to install both client and server on the Commerce nodes. Although WebSphere Commerce is technically an eXtreme Scale client, the WebSphere Commerce servers require the server installation because it contains the dynamic cache provider files.

With WebSphere eXtreme Scale version 7.1, you only need perform the client install on the Commerce nodes.

Warning: At the time of writing, there was an issue where the dynamic cache provider used the wrong grid name. This issue is resolved in WebSphere eXtreme Scale V7.1.0.1. The work around in the mean time is to move the following file to a temporary directory:

`WAS_HOME/lib/xsadmindynacachexc10.jar`

This file is only needed if using the dynamic cache provider from the DataPower XC10 appliance. If you are using the eXtreme Scale dynamic cache provider, the presence of this file causes a problem.

3. Augment the existing WebSphere Commerce profiles to enable the WebSphere eXtreme Scale functionality. These typically include the following profiles:
 - The deployment manager profile
This profile augmentation will enable the WebSphere eXtreme Scale-specific portions of the administrative console.
 - The custom profile(s) that contain the Commerce application servers
4. Create new WebSphere eXtreme Scale custom profiles for the WebSphere eXtreme Scale nodes and federate them to the deployment manager. For guidance on how many servers are needed, see 6.5, “Sizing guidance” on page 137.

We are now in a position to set up the WebSphere eXtreme Scale dynamic cache provider.

6.3.5 Sample environment configuration

The following sections describe in detail how to configure the various portions of the eXtreme Scale dynamic cache architecture:

- ▶ The clustered catalog servers in a catalog service domain
- ▶ The clustered eXtreme Scale dynamic cache grid
- ▶ The Commerce server configuration of the dynamic cache to use eXtreme Scale

Configuring the catalog server(s)

We are going to configure the catalog service domain part of our architecture, as shown in Figure 6-6 on page 111. The catalog server is needed to manage the placement of eXtreme Scale partitions and to allow clients to connect to the grid. By default, the WebSphere eXtreme Scale catalog server will run in the deployment manager of the cell. This is a convenient default, but is not resilient if it is the only catalog server. The catalog server can be configured to run in other servers too; such as the node agent(s), and application servers. Configure at least two or three catalog servers to run, ideally on separate physical servers.

We configured the catalog servers to run on the deployment manager and a single node agent by creating a catalog service domain. This is simply a grouping of local or remote catalog servers. When we define a catalog service domain of local catalog service endpoints we are doing two things: declaring where we want the catalog services to run, and providing the configuration so that eXtreme Scale grids and clients can connect to the eXtreme Scale catalog servers.

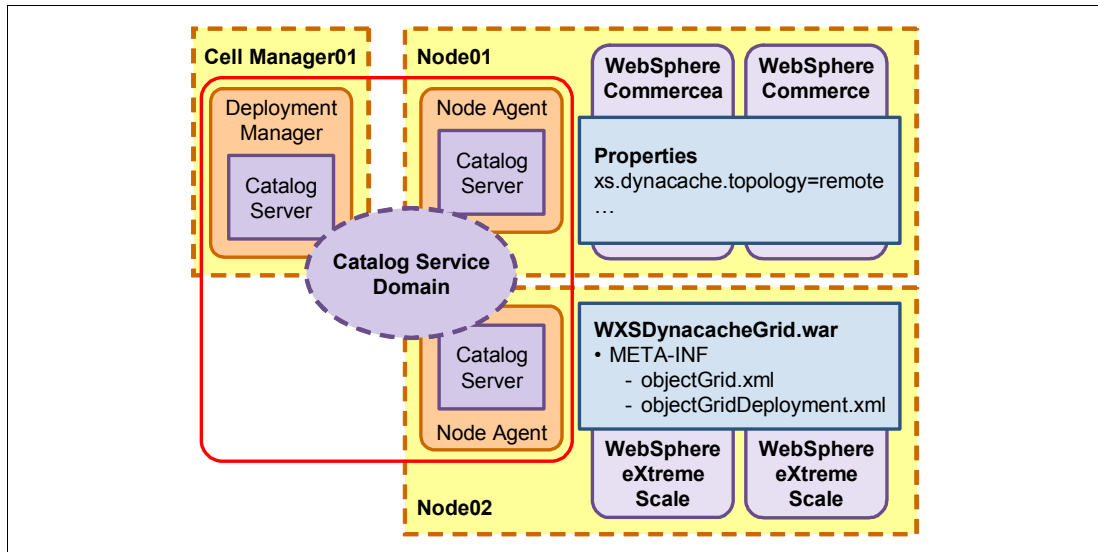


Figure 6-6 Configuring the catalog service domain

Determining catalog server ports: When configuring the catalog server ports, you need to provide port numbers for a client port, listener port, and JMX port. What you need to provide varies depending on whether the catalog server is hosted in a WebSphere application server or is running stand-alone.

For catalog servers that are hosted in WebSphere, both the listener port and JMX port are the same. They are simply the bootstrap port of application server. They are, by default, 9809 for the deployment manager and 2809 for the node agent.

The client port is used for eXtreme Scale internal communication and needs to be unique.

1. In the WebSphere administrative console (<http://hostname:9060/ibm/console>), click **System Administration** → **WebSphere eXtreme Scale** → **Catalog service domains**.
2. Click **New**.
3. Enter a name of your choice for your catalog server domain, for example:
Name: Clustered Catalog Servers
4. Select **Enable this catalog service domain as the default**.
5. Add the catalog servers to the catalog service domain, determining the ports as shown in Figure 6-7 on page 112.
 - a. Add a catalog server to the deployment manager:
 - i. A Catalog Service Endpoint will already exist. Select the deployment manager in the Existing application server drop down.
 - ii. Enter a client port value: 6600
 - iii. Enter a JMX port value: 9809

When hosting eXtreme Scale catalog servers in WebSphere Application Server, this port is the server's bootstrap address, which is 9809 by default on the deployment manager.
 - b. Add a catalog server to a node agent:
 - i. Click **New**.

- ii. Select a node agent from the list in the Existing application server drop down.
- iii. Enter a Client Port value: 6601

In the sample environment, the node agent is on the same hardware as the deployment manager, which means we need separate ports to make them unique

- iv. Enter a JMX Port value: 2809

The configuration window looks like Figure 6-7.

General Properties

* Name
Clustered Catalog Servers

☒ Enable this catalog service domain as the default unless another catalog service domain is explicitly specified.

JMX authentication credentials

Provide credentials for Java Management Extensions (JMX) authentication for retrieving status from the catalog server endpoints.

User ID
Password

Catalog Servers

New Delete

Select	Catalog Server Endpoint	Client Port	Listener Port	JMX Port
<input type="checkbox"/>	<input checked="" type="radio"/> Existing application server thinkCell01\thinkCellManager01\dmgr <input type="radio"/> Remote server	6600		9809
<input type="checkbox"/>	<input checked="" type="radio"/> Existing application server thinkCell01\thinkNode02\nodeagent <input type="radio"/> Remote server	6601		2809

Apply OK Reset Cancel

Figure 6-7 Create Catalog service domain

- c. Click **Apply**.

At this point the Listener Port field is disabled when adding existing servers. This is because WebSphere assumes the bootstrap port for the servers we are configuring, and so cannot change them.

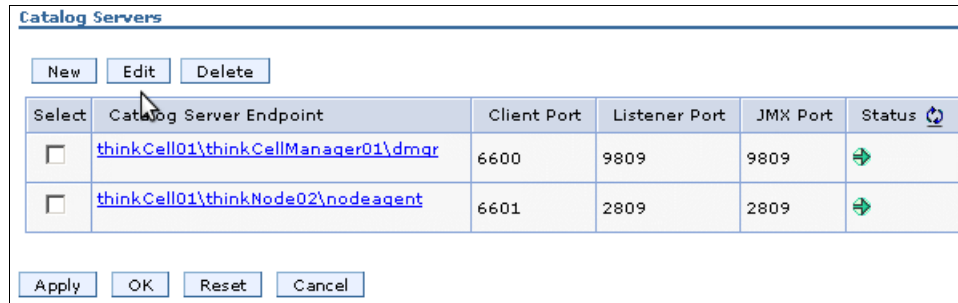
At the time of writing, however, if you are *not* using the default bootstrap ports (9809 for deployment managers and 2809 for node agents), these need to be added to the Listener Port field. Otherwise the grid will attempt to use the defaults. Even if you are using the defaults, you might choose to do this to add clarity in the configuration.

- d. On your new Catalog service domain, select the two catalog server entries and click **Edit**.
- e. Type in the listener port values appropriate to your environment. In the sample environment it is:

dmgr Listener Port = 9809

nodeagent Listener Port = 2809

The fully configured Catalog service domain is shown in Figure 6-8.



Select	Catalog Server Endpoint	Client Port	Listener Port	JMX Port	Status
<input type="checkbox"/>	thinkCell01\thinkCellManager01\dmgr	6600	9809	9809	➡
<input type="checkbox"/>	thinkCell01\thinkNode02\nodeagent	6601	2809	2809	➡

Figure 6-8 The fully configured catalog service domain (shown after restart)

6. Save the changes to the master configuration.
7. After changing catalog server settings, you will need to restart the servers that are to host the catalog servers: the deployment manager and node agent.

You can verify that the catalog server has started correctly by checking the `SystemOut.log` files for the deployment manager and node agent. You will see a message similar to the message in Example 6-1 indicating that the catalog server is starting.

Example 6-1 Log message to indicate that the catalog server is starting

```
[10/21/10 14:24:52:296 CDT] 0000000c ServerImpl I CW0BJ2518I: Launching  
ObjectGrid catalog service: thinkCell01\thinkNode02\nodeagent.
```

After both the deployment manager and node agent are started, they will form a catalog service cluster (one master catalog and one standby) and you will see a message similar to the message in Example 6-2.

Example 6-2 Log message to provide the status of the catalog server cluster

```
[10/21/10 14:25:37:343 CDT] 00000012 CatalogServer I CW0BJ8109I: Updated catalog  
service cluster CatalogCluster[thinkCell01, 1 master: 1 standbys] from server  
thinkCell01\thinkNode02\nodeagent with entry CatalogServerEntry...
```

Note: It is important to remember that because the deployment manager now hosts a catalog server, it is an important part of the runtime infrastructure. Prior to use with WebSphere eXtreme Scale, you could stop and start the deployment manager at will because it was not needed for the application servers. Stopping the deployment manager will now reduce catalog server availability, and therefore site availability.

Because we are running Commerce Server and eXtreme Scale in the same Network Deployment cell, we only need to set up one catalog service domain. But if we had one cell for Commerce and a separate cell for eXtreme Scale, we need to perform these steps twice:

1. Create a catalog service domain on the eXtreme Scale cell according to the steps we have outlined and define where the catalog servers are to run.
2. Create a catalog service domain on the Commerce cell. Instead of referring to existing servers in the drop down list, you need to provide details for the remote servers. You have to provide the following information for each catalog server defined in the eXtreme Scale Catalog service domain:

- Remote server: This is the host name of the remote catalog server.
- Listener port: This is the bootstrap port of the remote catalog server.
- JMX port: This is the JMX port number defined on the remote catalog server. When hosting the catalog server in WebSphere application servers, this value will be the same as the bootstrap port.

The administrative console doesn't need to know the client port for remote processes, so it doesn't prompt for it.

Standalone catalog servers: We have described the steps to set up and communicate with catalog servers running inside WebSphere processes. It is also feasible to run eXtreme Scale stand-alone, that is outside WebSphere processes. This is functionally identical to hosting the grid in WebSphere. The difference is that you need to manage the stand-alone eXtreme Scale processes manually.

The catalog service domain in WebSphere Commerce needs to be configured as we have outlined for connecting to a remote cell. If you want to connect to a remote stand-alone catalog server cluster, start that cluster beforehand using the `startOgServer` scripts. It should at minimum include the following parameters in order to configure the catalog service domain in the administrative console:

► `-listenerPort`

The catalog server bootstrap.

► `-JMXServicePort`

The port to access JMX MBeans. This defaults to 1099 if not specified.

► `-domain`

If you are using more than one catalog service domain, then you will need to specify a domain name for the catalog server cluster. Make this name the same as the name of the catalog service domain in the administrative console.

For more information about running eXtreme Scale stand-alone, see the Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/topic/com.ibm.websphere.extremescale.admin.doc/txscfgtcp.html>

We can now create an eXtreme Scale grid to hold our dynamic cache data and subsequently configure Commerce to use it.

Configure the WebSphere eXtreme Scale dynamic cache grid

The next step (as outlined in Figure 6-9 on page 115) is to configure the dynamic cache grid.

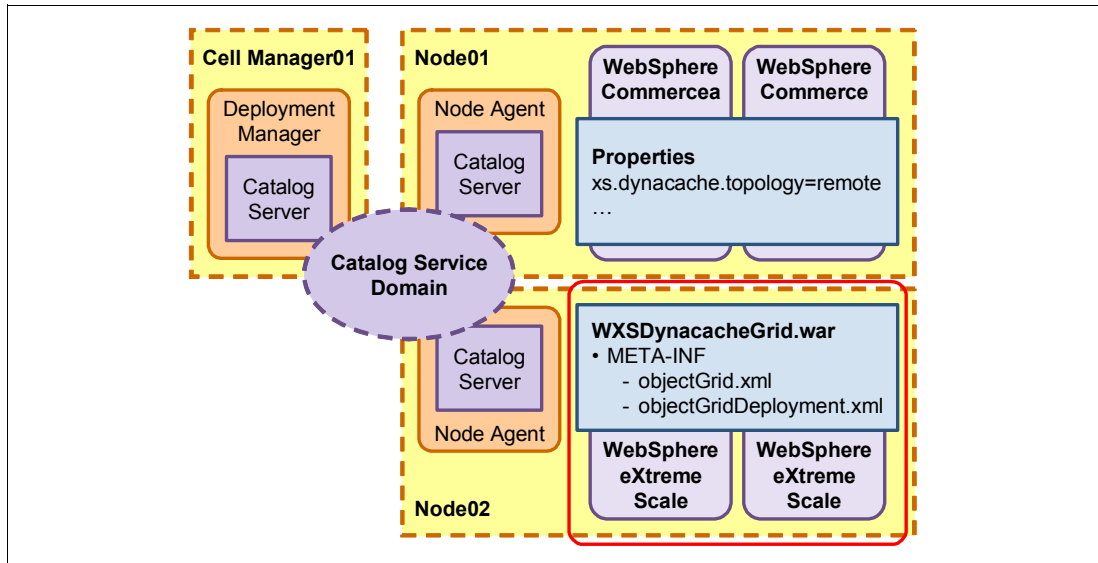


Figure 6-9 Configuring the clustered eXtreme Scale dynamic cache grid

Create the application server cluster

Create a WebSphere Application Server cluster of two application servers to host the dynamic cache data in an eXtreme Scale grid. We have chosen to create two application server members for simplicity in describing the steps required to configure a grid. In a production environment, use at minimum three cluster members to ensure an even spread of data in the grid. Sizing considerations are outlined in 6.5, “Sizing guidance” on page 137, and that will dictate how many application servers you need for the grid.

1. To create a new WebSphere Application Server cluster, in the administrative console, click **Servers** → **Clusters** → **WebSphere Application Server clusters** and click **New**.
2. Enter the cluster name, for example WXS Dynamic Cache Cluster, and click **Next**.
3. Add in the first application server cluster member with appropriate details:
 - a. Enter the member name: WXS_DC_ClusterMember1.
 - b. Select the node that the application server is going to run on.
 - c. Click **Next**.
4. Add additional application server cluster members. In this scenario, we created one additional application server, to have a total of two cluster members.
 - a. Enter the member name: WXS_DC_ClusterMember2.
 - b. Click **Add Member**.
 - c. Click **Next**.
5. Review the summary (as shown in Figure 6-10 on page 116) and click **Finish**.
6. Save your changes.

Create a new cluster

Step 1: Enter basic cluster information

Step 2: Create first cluster member

Step 3: Create additional cluster members

→ Step 4: Summary

Summary

Summary of actions:

Options	Values
Cluster Name	WXS Dynamic Cache Cluster
Core Group	DefaultCoreGroup
Node group	DefaultNodeGroup
Prefer local	true
Configure HTTP session memory-to-memory replication	false
Server name	WXS_DC_ClusterMember1
Node	thinkNode02(ND 7.0.0.11 WXS 7.1.0.0)
Weight	2
Clone Template	default
Clone Basis	Create the member using an application server template.
Generate unique HTTP ports	true
Server name	WXS_DC_ClusterMember2
Node	thinkNode02(ND 7.0.0.11 WXS 7.1.0.0)
Weight	2
Clone Template	Version 7 member template
Generate unique HTTP ports	true

Previous

Finish

Cancel

Figure 6-10 Summary of the new eXtreme Scale grid cluster

The application servers can be further tuned after the grid has been configured. Areas of configuration include the following parameters:

- The Java virtual machine heap size will be the primary tuning parameter to set. Set this at 1500 Mb as a starting point and tune as needed.
- Environment optimizations such as switching on the WebSphere Application Server version 7 feature “Start components as needed” (configured on the main application server configuration page in the administration console), which will not start internal components not used by the server, will leave more memory for eXtreme Scale.

Deploy the dynamic cache grid

The next step is to deploy the eXtreme Scale dynamic cache grid configuration. When you deploy the configuration for an eXtreme Scale grid to run on WebSphere Application Server, you need to package the eXtreme Scale configuration files into a web archive (WAR file) to deploy them. WebSphere eXtreme Scale provides a default configuration for the dynamic cache grid, which can be found in:

`WAS_HOME/optionalLibraries/ObjectGrid/dynacache/etc`

Grid versus ObjectGrid: ObjectGrid is the name of the technology within WebSphere eXtreme Scale, and was a previous name of the product. Hence it is used a lot in the configuration. The term “grid” used throughout this book in fact refers to an ObjectGrid.

Two files cover the definition of the grid and the nature of its deployment:

- ▶ `dynacache-remote-objectgrid.xml`
Contains the definition of the grid required by eXtreme Scale to host the dynamic cache data. This file should not need editing.
- ▶ `dynacache-remote-deployment.xml`
Contains deployment information for the grid. Review and edit this file.

To deploy the dynamic cache grid, perform the following steps:

1. Create a temporary folder location with a META-INF folder under it and copy the sample XML files into it before editing them. The files will need to be renamed (this is case sensitive) as shown in Figure 6-11. This is to adhere to the file name convention needed by eXtreme Scale for it to recognize the grid configuration in a WAR file.
 - Rename `dynacache-remote-objectgrid.xml` to `objectGrid.xml`
 - Rename `dynacache-remote-deployment.xml` to `objectGridDeployment.xml`

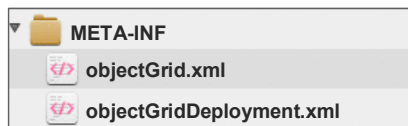


Figure 6-11 Required filenames and folder structure

2. Optional: Review the contents of `objectGrid.xml`. This is not provided here because it should not need editing. It contains the definition of the eXtreme Scale grid required to host the dynamic cache information.

Advanced option: It is possible to configure eXtreme Scale to run more than one dynamic cache grid. This can be used for hosting grids for more than one Commerce or other application server environment. To do this:

1. Create a separate set of `objectGrid.xml` and `objectGridDeployment.xml` files and rename `objectGrid` definition in the `objectGrid.xml` (for example, to `DYNACACHE_REMOTE_2`).
2. Edit `objectGridDeployment.xml` (shown in Example 6-3) to have the same `objectgridName` and `mapSet` name.
3. Add an additional JVM property (as added in step 3 on page 125) on Commerce JVMs to tell it to use a separate grid:
`com.ibm.websphere.xs.dynacache.grid_name`
4. Deploy the grid settings alongside other grids and register them with the same catalog server(s).

3. Review the contents of `objectGridDeployment.xml` as shown in Example 6-3.

Example 6-3 default objectGridDeployment.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="DYNACACHE_REMOTE">
```

```

    <mapSet name="DYNACACHE_REMOTE"
      numberOfPartitions="47"
      minSyncReplicas="0"
      maxSyncReplicas="0"
      maxAsyncReplicas="1"
      numInitialContainers="1"
      developmentMode="false"
      replicaReadEnabled="false">
      <map ref="IBM_DC_PARTITIONED_.*" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>

```

The key parameters to review and change where appropriate are:

- numInitialContainers=2

This value must be changed to equal the number of application servers in the eXtreme Scale grid cluster, in our example two. This prevents the grid starting until all application servers in the grid have started so it can balance the partitions evenly. Without this, there will be imbalance towards the initial application server started because eXtreme Scale will not move or balance partitions unless strictly necessary.

- numberOfPartitions=47

The default number of partitions is a good starting place. This number can be changed in line with sizing and capacity planning considerations (as discussed in more detail in 6.5, “Sizing guidance” on page 137).

- maxAsyncReplicas=1

WebSphere eXtreme Scale provides two modes of replication: synchronous and asynchronous. With the quality of service required by the dynamic cache service, asynchronous replication will be sufficient, and it naturally performs better than synchronous replication.

Set maxAsyncReplicas to the maximum number of asynchronous replicas you want. The default of 1 is probably sufficient and will mean that each partition will have one copy.

Zero is a viable option if you do not want to replicate the cache partitions. In this case, any cached data held on a stopped or failed JVM is lost and must be recreated. That might meet your cache availability requirement and saves memory by having no data duplication.

- developmentMode=false

This setting shouldn’t need changing but is commented on here for convenience. If testing the grid on a single server, then by default eXtreme Scale will not start replicas because it tries to put them on separate nodes from the primary for availability. For testing purposes, this can be set to true to test the impact of replication in a single node environment.

4. Create a web archive file (WAR) to deploy the eXtreme Scale configuration.

From your temporary directory, create a WAR file by running the **jar** command found in Example 6-4. If the **jar** command is not found, then provide a full path to the executable found in *WAS_HOME\java\bin*.

Example 6-4 jar command to create new web archive file

```
jar -cvf WXSdynacacheGrid.war *
```

```

added manifest
ignoring entry META-INF/
adding: META-INF/objectGridDeployment.xml(in = 668) (out= 354)(deflated 47%)
adding: META-INF/objectGrid.xml(in = 2132) (out= 722)(deflated 66%)

```

5. Deploy the WAR file to the WXS Dynamic Cache Cluster we created earlier. Deploy this as you deploy any normal application:
 - a. From the administrative console, select **Applications** → **New Application** → **New Enterprise Application**.
 - b. Click **Browse** for the Local file system option and select the WXSdynacacheGrid.war file that you just created. Click **Next**.
 - c. On Preparing for the application installation, select the default of Fast Path and click **Next**.
 - d. In the next five steps of the deployment, leave the defaults except for Step 2: Map modules to servers, where you will need to select the WXSdynacacheGrid.war module, select the WXS Dynamic Cache Cluster and click **Apply**, as shown in Figure 6-12.

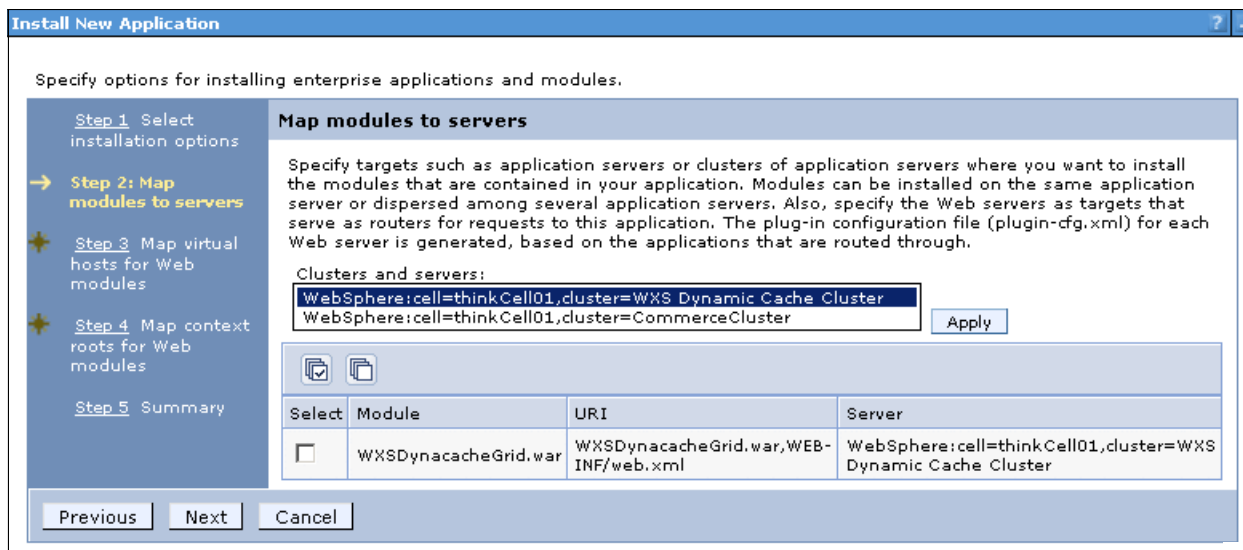


Figure 6-12 Map the application to the dynamic cache grid cluster

- e. On the summary page, click **Finish** and save your changes.
6. To start the eXtreme Scale grid, in the administrative console, click **Servers** → **Clusters** → **WebSphere application server clusters** → **WXS Dynamic Cache Cluster** and click **Start**.
7. When the cluster has started, check the cluster members' SystemOut.log files for messages that eXtreme Scale has started successfully.

When the application server itself is starting, you will see a message similar to that shown in Example 6-5. This shows that the product is correctly installed and being used by the cluster, and provides the exact product version.

Example 6-5 Log message to confirm successful detection of eXtreme Scale process

```

[10/22/10 13:19:41:468 CDT] 00000000 RuntimeInfo I CW0BJ0903I: The internal
version of WebSphere eXtreme Scale is v4.0 (7.1.0.0) WXS7.1.0.XS [a1025.57350].

```

Secondly, towards the end of the application server start-up, you can see where the web module starts to load. Example 6-6 shows that the eXtreme Scale configuration has been detected in the web module and the ObjectGrid Server is started. It will also reference the template map configuration from the objectGrid.xml file. A template is the prefix name for the dynamic cache grid(s) that will be created.

Example 6-6 Log messages to demonstrate grid start

```
[10/22/10 13:20:04:515 CDT] 00000019 ApplicationMg A WSVR0200I: Starting
application: WXSDynacacheGrid_war
[10/22/10 13:20:07:781 CDT] 00000019 ServerImpl I CWOBJ1001I: ObjectGrid
Server thinkCell01\thinkNode02\WXS_DC_ClusterMember1 is ready to process
requests.
[10/22/10 13:20:08:140 CDT] 00000019 XmlObjectGrid I CWOBJ4701I: Template map
info_server.* is configured in ObjectGrid IBM_SYSTEM_xsastats.server.
[10/22/10 13:20:08:250 CDT] 00000019 XmlObjectGrid I CWOBJ4701I: Template map
IBM_DC_PARTITIONED.* is configured in ObjectGrid DYNACACHE_REMOTE.
```

Finally, eXtreme Scale will wait until both application servers are fully started before actually starting the grid itself.

Tip: This is the importance of the `numInitialContainers` value in the `objectGridDeployment.xml` file. This value was set to 2 so that eXtreme Scale will wait for both servers to start.

You will see a message when each of the partitions and their replicas that are started, as illustrated in Example 6-7. A primary partition must always be on a separate cluster member than its replica.

Example 6-7 Log messages to show the eXtreme Scale partitions and replicas activating

```
...
[10/29/10 4:24:19:896 CDT] 00000021 AsynchronousR I CWOBJ1511I:
DYNACACHE_REMOTE:DYNACACHE_REMOTE:10 (asynchronous replica) is open for
business.
[10/29/10 4:24:19:896 CDT] 00000021 AsynchronousR I CWOBJ1543I: The
asynchronous replica DYNACACHE_REMOTE:DYNACACHE_REMOTE:10 started or continued
replicating from the primary. Replicating for maps:
[xsastats_DYNACACHE_REMOTE_grid, xsastats_DYNACACHE_REMOTE_map]
[10/29/10 4:24:22:802 CDT] 00000038 ReplicatedPar I CWOBJ1511I:
DYNACACHE_REMOTE:DYNACACHE_REMOTE:15 (primary) is open for business.
...
```

8. Verify the grid deployment and grid access.

WebSphere eXtreme Scale provides a command line tool, called **xsadmin**, for querying the eXtreme Scale runtime configuration. Here are simple commands for validating and reviewing the configuration.

Warning: At the time of writing, there is an issue running **xsadmin** from the installation of eXtreme Scale v7.1 when installed on WebSphere Application Server v7.0, where it wouldn't connect properly to the catalog server. The work around is to use **xsadmin** from a separate, stand-alone installation of WebSphere eXtreme Scale.

The problem is resolved in WebSphere eXtreme Scale V7.1.0.1. and WebSphere eXtreme Scale V6.1.0.5. The resolution is also available as a patch at:

<http://www-01.ibm.com/support/docview.wss?uid=swg1PM20613>

From the deployment manager profile bin directory, run the **xsadmin** commands with a **-dmgr** flag, telling **xsadmin** that it is connecting to a catalog server in a deployment manager and to use the default deployment manager ports to connect. If your deployment manager is running on a non-default bootstrap port, then specify that with the **-p** option.

Example 6-8 shows a list of all running grids and maps. You will see that it correctly lists our DYNACACHE_REMOTE grid.

Example 6-8 Output from xsadmin utility to list grids and maps

```
C:\<WAS_HOME>\profiles\Dmgr01\bin>xsadmin.bat -dmgr -l
```

This administrative utility is provided as a sample only and is not to be considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:9809

Warning: Detected there is more than one Placement Service MBean

*** Show all 'objectGrid:mapset' names

Grid Name	MapSet Name
DYNACACHE_REMOTE	DYNACACHE_REMOTE
IBM_SYSTEM_xsastats.server	stats

Example 6-9 shows a list of all containers (JVMs) and their partitions. This lists all 47 partitions and their replicas shared evenly between the two cluster members.

Example 6-9 Output from xsadmin utility to list the containers

```
C:\<WAS_HOME>\profiles\Dmgr01\bin>xsadmin.bat -dmgr -containers
```

This administrative utility is provided as a sample only and is not to be considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:9809

Warning: Detected there is more than one Placement Service MBean

*** Show all online containers for grid - DYNACACHE_REMOTE & mapset - DYNACACHE_REMOTE

Host: think.was7.ibm.com

Container: thinkCell101\thinkNode02\WXS_DC_ClusterMember1_C-1,

Server:thinkCell101\thinkNode02\WXS_DC_ClusterMember1, Zone:DefaultZone

Partition	Shard Type
0	AsynchronousReplica
1	AsynchronousReplica
2	AsynchronousReplica
8	AsynchronousReplica
... 40	Primary
42	Primary

```

43      Primary
Container: thinkCell01\thinkNode02\WXS_DC_ClusterMember2_C-1,
Server:thinkCell01\thinkNode02\WXS_DC_ClusterMember2, Zone:DefaultZone
Partition Shard Type
3      AsynchronousReplica
4      AsynchronousReplica
5      AsynchronousReplica
... 44      Primary
45      Primary
46      Primary

Num containers matching = 2
Total known containers  = 2
Total known hosts       = 1

```

We have now created an application server cluster that is hosting an eXtreme Scale grid that we can use to store dynamic cache data from WebSphere Commerce. We have verified that the grid has been deployed and started successfully.

All of these steps involve hosting the eXtreme Scale grid within WebSphere Application Server. It is possible to run the eXtreme Scale grid outside WebSphere on stand-alone JVMs. This will function in exactly the same way and the eXtreme Scale grid configuration is the same. The difference is that the stand-alone deployment will not have any management tooling for grid deployment and operational control. In the case of a stand-alone grid, the XML descriptors do not need packaging but can be referred to directly from the command line. For more information about running stand-alone eXtreme Scale containers, see the Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/txssastartstop.html>

Configure the WebSphere Commerce application servers

We are now ready to configure the WebSphere Commerce servers to use eXtreme Scale for storing dynamic cache content by setting properties in the Commerce application servers, as highlighted in Figure 6-13 on page 123. As outlined earlier, for this we are going to assume that there is an existing WebSphere Commerce server or cluster that we are going to configure.

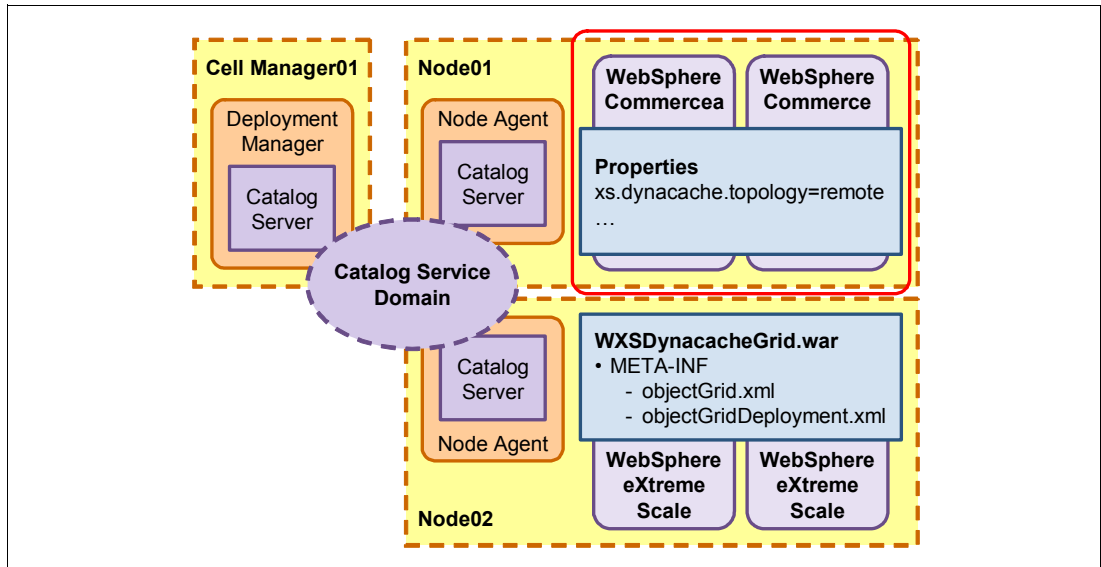


Figure 6-13 Configuring the Commerce servers to use eXtreme Scale for the dynamic cache provider

Typically, WebSphere Commerce will already be configured to use the default dynamic cache service, but for completeness we are going to outline the steps to set up the dynamic cache from the beginning to use eXtreme Scale.

Perform the following steps for each Commerce application server.

1. Enable servlet caching for the application server to switch on dynamic web caching. This is probably already set by default in Commerce application servers.
 - a. In the WebSphere administrative console, click **Servers** → **Server Types** → **WebSphere application servers** and select your Commerce application server.
 - b. On the right of the application server page, click **Container Settings** → **Web Container Settings** → **Web container**.
 - c. Select **Enable servlet caching**, as shown in Figure 6-14.

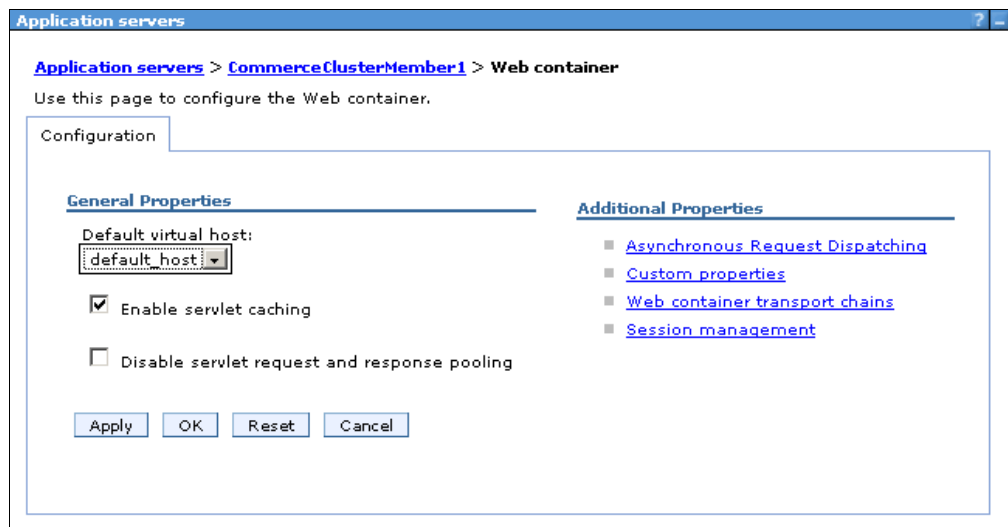


Figure 6-14 Enable servlet caching

- d. Click **OK** and save the changes.

2. Set up the dynamic cache service to use WebSphere eXtreme Scale:
 - a. On the right of the Commerce application server configuration page, select **Container Services** → **Dynamic Cache Service**.
 - b. In the drop down, select **WebSphere eXtreme Scale** as shown in Figure 6-15 on page 125. This will change the dynamic cache provider for the baseCache (the default cache instance) to use eXtreme Scale.
 - c. Leave the cache size as 2000 entries for now. As mentioned earlier, this is the number of entries that WebSphere eXtreme Scale will store *per partition* configured on the grid. For guidance on setting the cache size, see 6.5, “Sizing guidance” on page 137.
 - d. Ensure that the **Enable cache replication** option is enabled.

Avoid trouble: The **Enable cache replication** setting is not just used for the WebSphere Application Server dynamic cache. It is important that it is set for use with eXtreme Scale.

If you do not enable this setting, the eXtreme Scale configuration that we are about to set to dictate the topology (in step 3 on page 125) will be ignored. Instead, you will actually get a simple local eXtreme Scale grid in each Commerce server. This is similar to the traditional dynamic cache provider, but a little faster.

When you enable this setting, the rest of the eXtreme Scale configuration will be applied and, in this scenario, your remote grid will be used.

- e. Click **OK** and save the changes.

Using previous versions: The steps outlined in 2) are specific to WebSphere Application Server and Commerce version 7, where we use newer elements of the administrative console to configure the cache. This functionality is available in WebSphere Application Server version 6.1, but is done through the following JVM properties:

```
com.ibm.ws.cache.CacheConfig.cacheProviderName=  
    com.ibm.ws.objectgrid.dynacache.CacheProviderImpl  
  
com.ibm.ws.cache.CacheConfig.enableCacheReplication = true
```

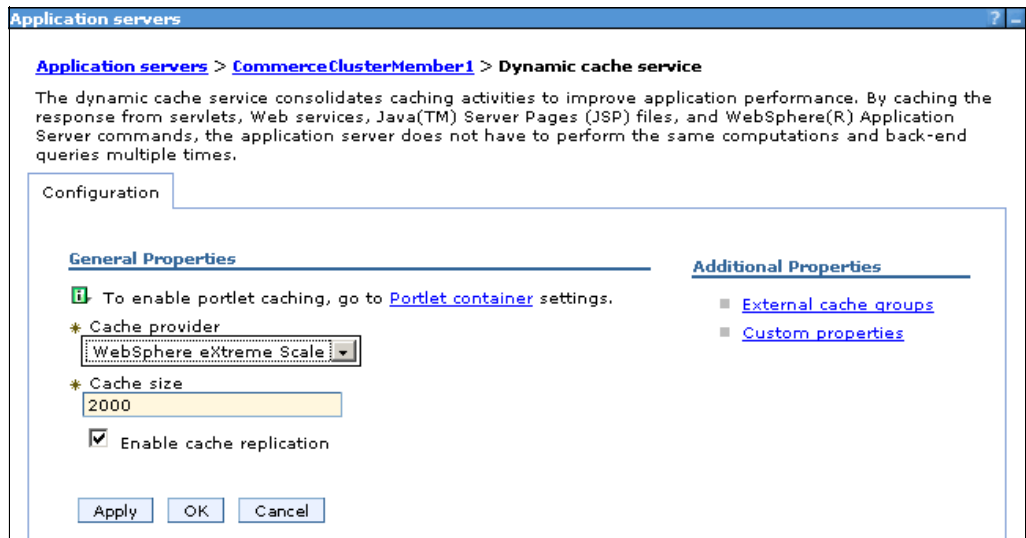


Figure 6-15 Change the dynamic cache provider

3. Set the appropriate dynamic cache properties.

There are a number of properties to set for the dynamic cache provider. These are both for the dynamic cache service and the eXtreme Scale provider. The only property that is actually required is the property that designates the eXtreme Scale provider topology. The other properties are recommended performance settings.

If you already have the dynamic cache service configured for use with Commerce, you probably have many properties already set. It is worth pointing out that any properties that you have set are not used by the eXtreme Scale provider. They can be safely ignored. This will allow you to easily revert back to the default dynamic cache provider if you want to test and compare the behavior with eXtreme Scale.

There is little in the way of additional configuration in eXtreme Scale because dynamic cache properties are translated to reasonable settings for eXtreme Scale.

For further information as to what properties do and do not apply to eXtreme Scale, see the WebSphere Application Server Information Center at the following address for comparison:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/udyn_rcachesettings.html

- On the right of the Commerce application server configuration page, select **Java and Process Management** → **Process definition** → **Java Virtual Machine**.
- Click **Custom properties**.
- For each of the following properties, click **New** and enter the name and value for each property.

Table 6-1 Performance properties for the dynamic cache service

Custom Property Name	Value	Description
com.ibm.websphere.xs.dynacache.topology	remote	This is the topology we are deploying to. Options include embedded, embedded_partitioned, and remote.

Custom Property Name	Value	Description
com.ibm.websphere.xs.dynacache.enable_compression	true	This is a performance recommendation to compress the content of the dynamic cache, with the benefits of reduced data to cache and reduced network traffic.
com.ibm.websphere.xs.dynacache.ignore_value_in_change_event	true	This is a performance recommendation where “true” means the cache entry is not de-serialized on the occurrence of a change event.
com.ibm.websphere.xs.dynacache.disable_recursive_invalidate	true	This is a performance recommendation to prevent recursive invalidations of dependency ids. This is a change in behavior, but for a feature rarely used.
com.ibm.ws.cache.CacheConfig.filterLRUInvalidation	true	This property is a dynamic cache recommendation and not specific to eXtreme Scale. It prevents the creation of invalidation events in the case of least recently used (LRU) eviction, which can cause large object allocations. More information about this setting can be found at: http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK24146
com.ibm.ws.cache.CacheConfig.filterTimeoutInvalidation	true	This property is a dynamic cache recommendation and not specific to eXtreme Scale. It prevents the creation of invalidation events in the case of time out eviction, which can cause large object allocations. More information about this setting can be found at: http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK24146
com.ibm.ws.cache.CacheConfig.disableTemplateInvalidation	true	This is a performance setting recommended to prevent invalidation based on cache templates, which is more expensive in eXtreme Scale.
com.ibm.ws.cache.CacheConfig.useServerClassLoader	true	This is a performance recommendation to specify which class loader to use to (de)serialize event information instead of dynamically having to determine the class loader. Commerce provides classes for this and therefore uses the server class loader. More information about this setting can be found at: http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK15310

The results are shown in Figure 6-16 on page 127.

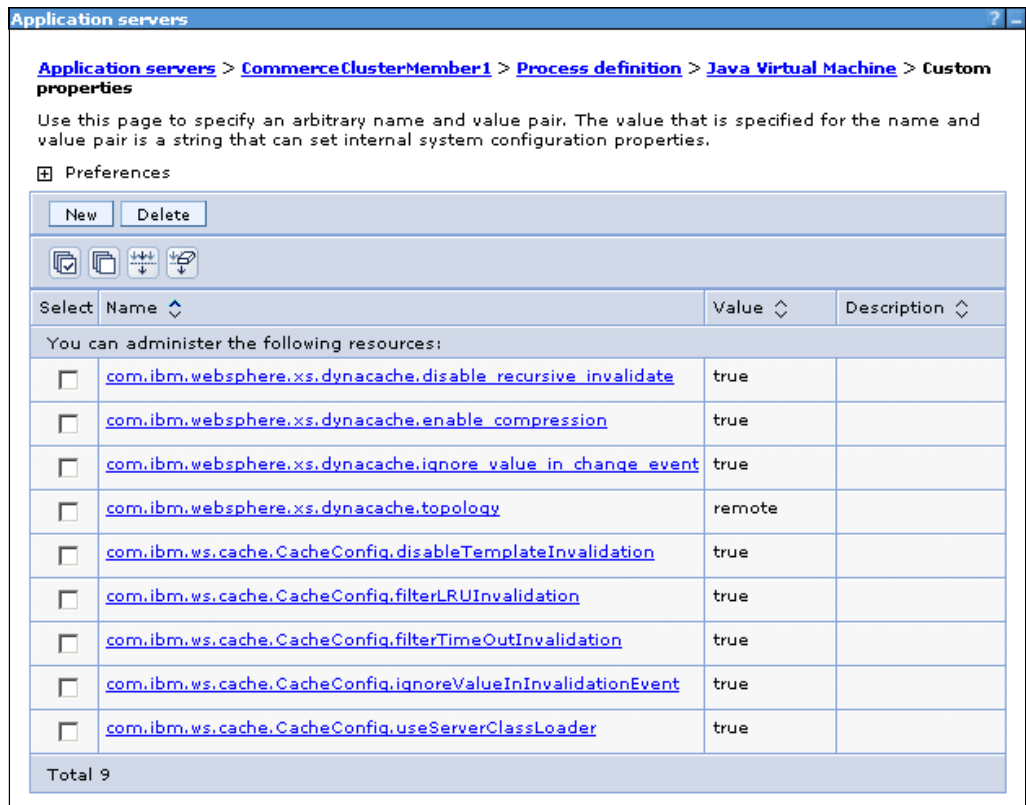


Figure 6-16 Recommended properties for the eXtreme Scale dynamic cache provider

- d. Repeat steps a on page 125 to c on page 125 for the other Commerce application servers.

Tip: You can save yourself having to type in lots of properties many times by fully configuring the first application server and then using the WebSphere template functionality to save it. In the administrative console,

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. From that page, select your configured application server and click **Templates....**
3. Select the Commerce application server that you have configured and provide a template name.

This template can be used to create new servers or clusters with the properties and dynamic cache provider already configured.

- e. Save your changes.

A full break-down of what settings directly apply to WebSphere eXtreme Scale and apply just to the default provider can be found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/udyn_rcachesettings.html

At this point, the Commerce application servers have the eXtreme Scale dynamic cache provider fully configured for the baseCache cache instance. You might want to test that this is properly configured and works at this stage (see “Start the environment” on page 130).

There are a few WebSphere Commerce-specific optimizations that we will make in the next section before the environment is completely ready.

Configure the data caches

All of the configuration so far actually applies to any WebSphere Application Server use of the dynamic cache. Now we are going to consider WebSphere Commerce-specific tuning. So far you have configured all the dynamic caching, by default, to be stored in WebSphere eXtreme Scale. This is not actually the best place for *all* Commerce cached data. WebSphere eXtreme Scale is a remote cache, but WebSphere Commerce is assuming all data is local, so it makes frequent access to certain types of data, such as user data. Placing these small but frequently accessed pieces of data into WebSphere eXtreme Scale impacts performance, so set up your environment to continue to use the default dynamic cache provider for those.

The WebSphere Commerce data caches use dynamic cache APIs (as introduced in 6.2, “Overview of caching differences with eXtreme Scale” on page 100.) to store output from expensive commands and store frequently accessed data. For more information about data caches, go to the following address:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.admin.doc/tasks/tdccdcont.htm>

Table 6-2 shows where you store dynamic cache data.

Table 6-2 Where to put cached data

Data cache type	Best cache environment
baseCache This is the default servlet cache, which we have already configured. Assume all web content fits into this category	WebSphere eXtreme Scale
Command caching	WebSphere eXtreme Scale
Object caches (for DistributedMap data)	Default dynamic cache provider

Command caching

A command data cache stores the response from expensive business logic methods. These caches are enabled by providing a `cachespec.xml` file that defines which command responses to cache and for how long. WebSphere Commerce provides a range of command caching configuration, with an example `cachespec.xml` file that can be found in:

`Commerce_installdir\samples\dynacache\cachespec.xml`

Because this configuration file does not define where it is going to store data, it will use the object cache called `default`, which is automatically configured on an application server.

Because you set the dynamic cache provider on the Commerce application server to use WebSphere eXtreme Scale, the `default` object cache is already configured to use eXtreme Scale as the repository and no other configuration is needed (See “Configure the WebSphere Commerce application servers” on page 122 for more information).

If there are additional command caches that need to be created, they can be configured in the administrative console (under **Resources** → **Cache instances** → **Object Cache instances**) or by deploying a `cacheinstances.properties` file with the application. An example of an object cache instance configuration in a `cacheinstances.properties` file can be found in Example 6-10 on page 129, which shows how the WebSphere eXtreme Scale provider is specified.

Example 6-10 Sample for additional command cache instances

```
cache.instance.0=/services/cache/SampleCache
cache.instance.0.cacheSize=2000
cache.instance.0.disableDependencyId=false
cache.instance.0.enableCacheReplication=true
cache.instance.0.filterLRUInvalidation=true
cache.instance.0.filterTimeOutInvalidation=true
cache.instance.0.disableTemplatesSupport=true
cache.instance.0.ignoreValueInInvalidationEvent=true
cache.instance.0.useServerClassLoader=true
cache.instance.0.cacheProviderName=com.ibm.ws.objectgrid.dynacache.CacheProviderImpl
cache.instance.0.com.ibm.websphere.xs.dynacache.disable_recursive_invalidate=true
cache.instance.0.com.ibm.websphere.xs.dynacache.ignore_value_in_change_event=true
cache.instance.0.com.ibm.websphere.xs.dynacache.enable_compression=true
cache.instance.0.com.ibm.websphere.xs.dynacache.topology=remote
```

Object caches

The data caches that use the DistributedMap APIs are used for storing Java objects. The object cache instances can be configured in the administrative console (under **Resources** → **Cache instances** → **Object Cache instances**) or by deploying a cacheinstances.properties file with the application.

Store the DistributedMap data in the default dynamic cache provider. These data caches can therefore be configured as they would be without WebSphere eXtreme Scale. An example of this is shown in Example 6-11.

Example 6-11 Sample for object caches using the default dynamic cache provider

```
cache.instance.0=/services/cache/WCSystemDistributedMapCache
cache.instance.0.cacheSize=1000
cache.instance.0.enableDiskOffload=false
cache.instance.0.flushToDiskOnStop=false
cache.instance.0.useListenerContext=false
cache.instance.0.replicationType=4
cache.instance.0.disableDependencyId=false
cache.instance.0.filterTimeOutInvalidation=true
cache.instance.0.filterLRUInvalidation=true
cache.instance.0.ignoreValueInInvalidationEvent=true
cache.instance.0.disableTemplatesSupport=true
cache.instance.0.useServerClassLoader=true
cache.instance.0.enableCacheReplication=true
cache.instance.1.replicationDomain=DataCacheRepDomain
```

Alternatively, the same configuration can be done in the administrative console. For example, the Marketing object cache instances are shown in Figure 6-17.

<div>NewDelete</div>				
<div><div></div><div></div><div></div><div></div></div>				
Select	Name	JNDI name	Scope	Cache size
You can administer the following resources:				
<input type="checkbox"/>	MarketingCache	services/cache/DM_Cache	Cluster=WXS_cluster	2500
<input type="checkbox"/>	MarketingUserCache	services/cache/DM_UserCache	Cluster=WXS_cluster	10000

Figure 6-17 The Marketing Object Cache instances

Simply click **New** to define and configure a new object cache instance.

Figure 6-18 shows a sample configuration for setting an object cache to use the default dynamic cache provider. The values shown are for illustration only, but demonstrate how to ensure the cache provider is set.

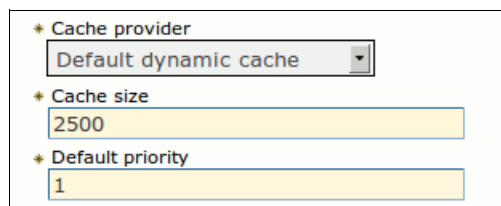


Figure 6-18 Defining the cache provider

Replication domain settings (found on the same configuration page as the cache provider settings) can be used in the standard way as illustrated in Figure 6-19.

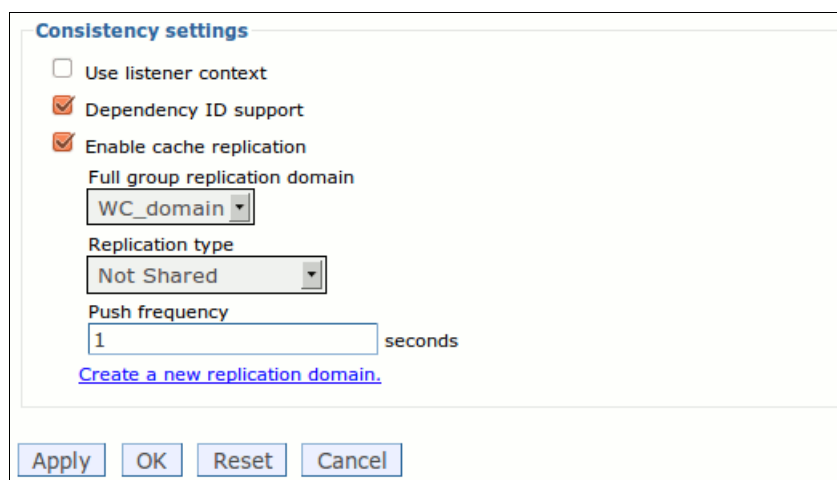


Figure 6-19 Replication settings

Start the environment

At this point, you have configured the whole environment and have validated that the catalog servers and eXtreme Scale grid are working correctly. Now you need to validate that the Commerce application servers are correctly using the eXtreme Scale dynamic cache provider and validate that the whole environment works.

Figure 6-5 on page 108 shows an operational overview of the environment. This illustrates the components you need to start and validate.

Start order: The order in which the application server components are started is important. Ensure you start the components in the following order:

1. The catalog server components; namely the deployment manager and node agent(s)
2. The eXtreme Scale application server cluster
3. The Commerce application server cluster

Once started, the environment is flexible concerning the starting and stopping of components. For example, the environment will be resilient if a single application server or catalog server fails.

After the catalog server components (deployment manager and node agent(s)) and eXtreme Scale are started, start the Commerce application server cluster. You can verify that it is using eXtreme Scale by looking in the `SystemOut.log` file for the Commerce application server as shown in Example 6-12.

Example 6-12 SystemOut.log showing eXtreme Scale as the Commerce dynamic cache provider

```
[10/25/10 8:46:08:538 CDT] 00000000 CacheProvider I   CWOBJ4500I: WebSphere eXtreme Scale
Dynamic Cache provider is successfully initialized.
[10/25/10 8:46:09:913 CDT] 00000000 ObjectGridImp I   CWOBJ4700I: The map name
IBM_DC_PARTITIONED_baseCache matched the regular expression of template map
IBM_DC_PARTITIONED_.*. The IBM_DC_PARTITIONED_baseCache map has been created for
ObjectGrid DYNACACHE_REMOTE.
[10/25/10 8:46:09:991 CDT] 00000000 CacheProvider I   CWOBJ4508I: The WebSphere eXtreme
Scale provider has created a Dynamic Cache instance with name baseCache using topology
remote.
```

Troubleshooting

If the eXtreme Scale grid is *not* available when starting the Commerce application servers, the Commerce servers will still successfully start. However, because they cannot use the eXtreme Scale grid, they will revert to using the default dynamic cache provider instead. This can be verified in the log file upon Commerce server start as shown in Example 6-13.

Example 6-13 SystemOut.log showing the Commerce server reverting to the default dynamic cache

```
[11/26/10 14:32:37:324 CST] 00000000 CacheProvider E   CWOBJ4501E: The WebSphere eXtreme
Scale Dynamic Cache provider encountered an error while creating the following cache
instance: baseCache.
[11/26/10 14:32:37:449 CST] 00000000 CacheProvider W   CWOBJ0006W: An exception occurred:
com.ibm.websphere.objectgrid.ObjectGridRuntimeException: The catalog server did not contain
an ObjectGridDeployment matching the name DYNACACHE_REMOTE
    at
com.ibm.ws.objectgrid.ObjectGridManagerImpl.getRemoteObjectGridDeployment(ObjectGridManager
Impl.java:440)
...
[11/26/10 14:32:37:449 CST] 00000000 ServerCache   E   DYNA1066E: Unable to initialize the
cache provider "com.ibm.ws.objectgrid.dynacache.CacheProviderImpl". The Dynamic cache will
be used to create the cache instance "baseCache" instead of the configured cache provider.
[11/26/10 14:32:37:449 CST] 00000000 ServerCache   E   ENGLISH ONLY MESSAGE: coreCache ==
null || featureSupport == null....Check FFDC logs for Exceptions
[11/26/10 14:32:37:480 CST] 00000000 ServerCache   I   DYNA1001I: WebSphere Dynamic Cache
instance named baseCache initialized successfully.
[11/26/10 14:32:37:480 CST] 00000000 ServerCache   I   DYNA1071I: The cache provider
"default" is being used.
```

It is also possible to verify that the eXtreme Scale grid is being used through the Extended Cache Monitor (introduced in 6.4.1, “The Cache Monitor” on page 133), in which you can view MBeans specific to eXtreme Scale to confirm it is correctly being used.

Validate the caching configuration

In our sample environment, we used a simple web page to test the caching behavior of the configuration. See Example 6-14 on page 132 for a simple dynamically-generated web page. If the caching is working, the time on the page will not update when you refresh the web page.

Example 6-14 Test.jsp to verify dynamic cache behavior

```
<html><head><title>Dynacache Tester</title></head><body>

<H1>Test JSP</H1>
The time is <%=new java.util.Date().toString() %> <BR>
If the time does <b>not</b> update upon page refresh then the dynacache is working
correctly<BR>
</body></html>
```

See Example 6-15 for the corresponding cachespec.xml.

Example 6-15 cachespec.xml deployed in the WEB-INF directory of the web module

```
<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
  <cache-entry>
    <class>servlet</class>
    <name>/Test.jsp</name>
    <cache-id>
      <component id="*" type="parameter">
        <required>false</required>
      </component>
      <timeout>180</timeout>
    </cache-id>
  </cache-entry>
</cache>
```

The web page can be executed with a number of arbitrary parameters to create cache entries. For example, the following will create a new cache entry each:

```
http://localhost:9080/DynacacheTestWeb/Test.jsp?anyparam=anyvalue1
http://localhost:9080/DynacacheTestWeb/Test.jsp?anyparam=anyvalue2
http://localhost:9080/DynacacheTestWeb/Test.jsp?anyparam=anyvalue3
```

The caching statistics of the Test.jsp can then be reviewed in the appropriate monitoring tools introduced in 6.4, “Monitoring considerations” on page 132.

6.4 Monitoring considerations

There are a range of tools shipped with the WebSphere products for basic monitoring. WebSphere eXtreme Scale does add additional monitoring tools. However, the primary tools that you use today with WebSphere Commerce continue to apply after integrating WebSphere eXtreme Scale. eXtreme Scale in most cases simply augments the data available. The primary tools to use are:

- ▶ The Cache Monitor: for dynamic cache monitoring
- ▶ The Tivoli Performance Viewer: for application server related metrics
- ▶ xsadmin: for eXtreme Scale deployment monitoring

These cover the basic monitoring requirements of a WebSphere Commerce caching infrastructure; cache management, cache performance, and cache health. Other options will be discussed that can be useful to supplement existing monitoring approaches and tools.

6.4.1 The Cache Monitor

The Cache Monitor is a fully supported dynamic cache management application that comes with WebSphere Application Server and can be installed from the `WAS_HOME/installableApps` directory.

The Cache Monitor application is the primary tool provided to allow administrators to manage the dynamic cache environment. It provides the following features:

- ▶ Cache statistics such as hit rates and cache size
- ▶ Displays the contents of the cache including cache keys and value dependencies
- ▶ Manual invalidation control over the cache contents
- ▶ Cache configuration details such as templates and dependencies

This tool is specific to the dynamic cache APIs and works regardless of whether you're using WebSphere eXtreme Scale or not. However the context of the values will be different if you are using WebSphere eXtreme Scale. When you are using the default dynamic cache provider, you need to install the Cache Monitor application on each server you want to monitor to get statistics specific to that server. When you use WebSphere eXtreme Scale, the values refer to the eXtreme Scale grid as a whole. With both approaches, these statistics are not synchronized and therefore can vary up to 10% for concurrent workloads.

Warning: At time of writing there is an issue that might cause deadlock on the eXtreme Scale grid when trying to perform a full-cache clear. The problem is resolved in WebSphere eXtreme Scale V7.1.0.1. The resolution is also available as a patch at:

<http://www-01.ibm.com/support/docview.wss?uid=swg1PM21272>

When using WebSphere eXtreme Scale with the dynamic cache, consider using the technology preview called the Extended Cache Monitor. This is freely available from the developerWorks site at the following address:

http://www.ibm.com/developerworks/websphere/downloads/cache_monitor.html

The Extended Cache Monitor, shown in Figure 6-20 on page 134, is just an extension installed on top of the supplied Cache Monitor and works in the same way. The Extended Cache Monitor is better suited to managing multiple dynamic cache providers, such as the scenario we recommend here using eXtreme Scale for web content and the default dynamic cache provider for data caches. It also provides a selection of MBean statistics and allows the viewing of object cache contents.

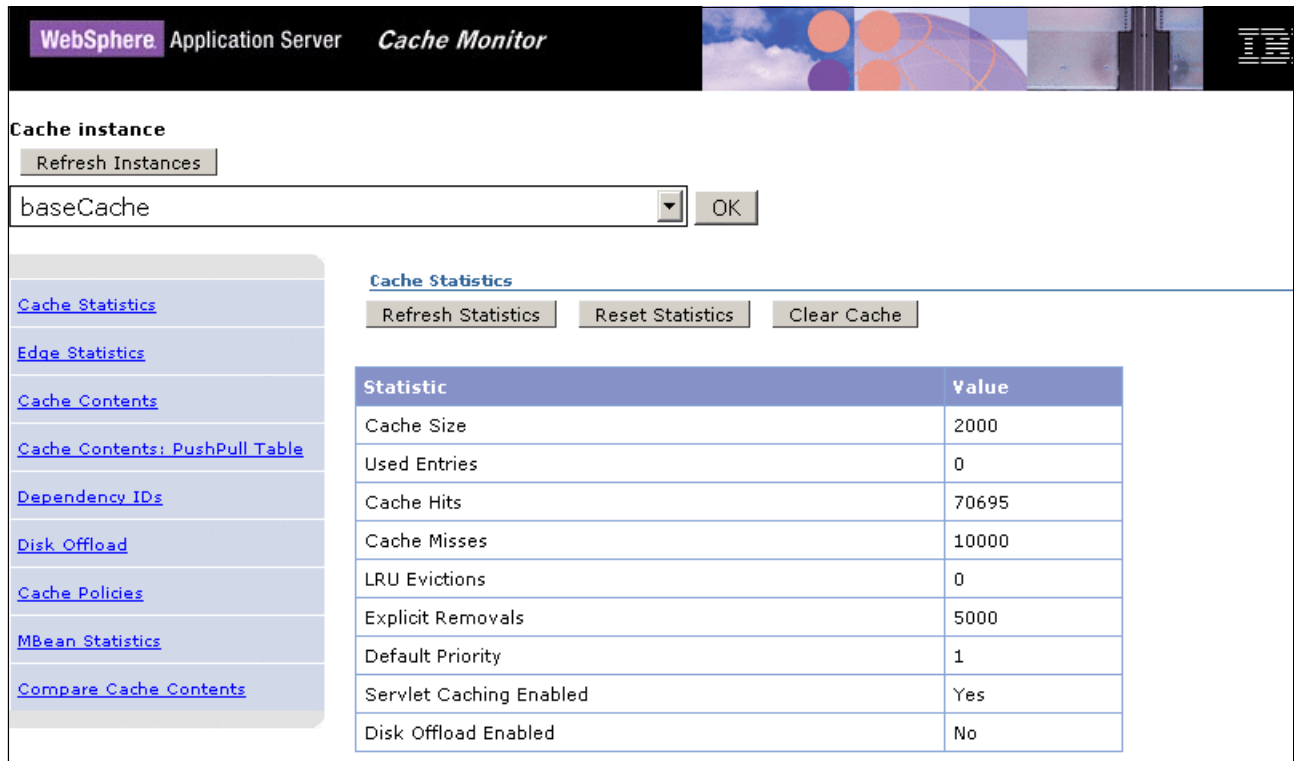


Figure 6-20 Extended Cache Monitor: Cache statistics

The Cache Monitor and Extended Cache Monitor are simple to set up and use, but if further information is needed, see the Information Center at the following address:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.webSphere.nd.multiplatform.doc/info/ae/ae/tdyn_servletmonitor.html

6.4.2 The Tivoli Performance Viewer

You can enable the performance monitoring infrastructure of WebSphere Application Server to collect metrics relevant to WebSphere eXtreme Scale. These metrics provide information about response times and the cache. The Tivoli Performance Viewer, available in the WebSphere administrative console, can be used to view these metrics.

Figure 6-21 on page 135 shows an example of the metrics for the ObjectGrid maps collected for one of the servers in the eXtreme Scale cluster.

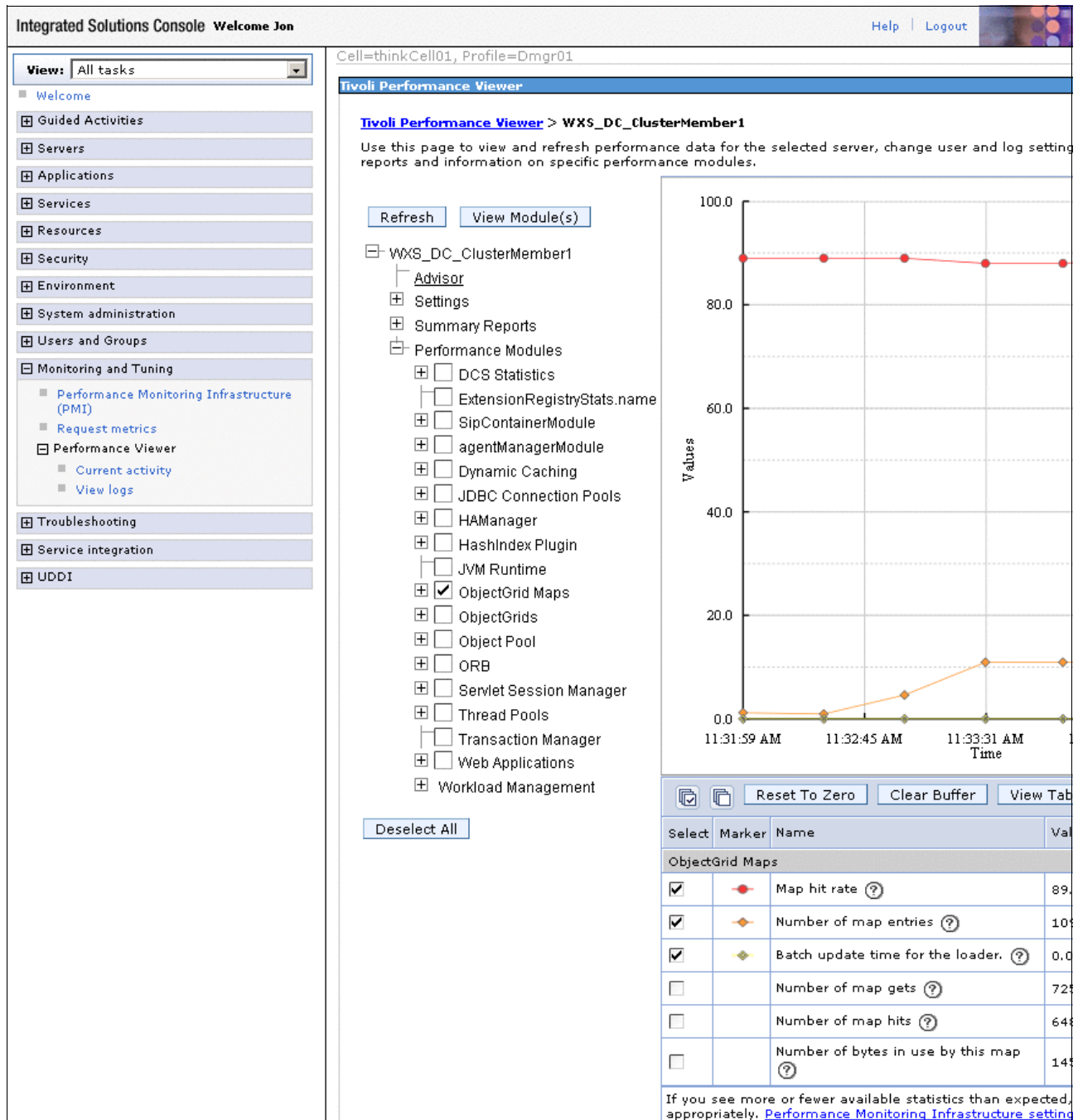


Figure 6-21 Performance Viewer: ObjectGrid Map statistics

The primary benefit of using the Performance Viewer is when you need to analyze the performance and behavior of an application server. You can review the eXtreme Scale statistics in the context of the other application server metrics.

Information on setting up the Tivoli Performance Viewer for use with eXtreme Scale can be found in 3.8.3, “Monitoring with the Tivoli Performance Viewer” on page 50. More information can be found in the Information Center at the following address:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/tpvf_tpvmonitor.html

6.4.3 xsadmin

WebSphere eXtreme Scale provides a command line tool to query the eXtreme Scale runtime deployment configuration. We mentioned **xsadmin** previously when we wanted to validate that our eXtreme Scale environment was properly running (“Configure the WebSphere eXtreme Scale dynamic cache grid” on page 114). The primary purposes of **xsadmin** are to verify that the grid configurations are running, verify where partitions and replicas are running, and to query map sizes.

For example, use the **xsadmin** command to list running partitions with their corresponding sizes in terms of number of entries and used heap in KB. The command used is:

xsadmin -dmgr -mapsizes

The entire **xsadmin** session is shown in Example 6-16.

Example 6-16 Partial output from the xsadmin -dmgr -mapsizes command (only 5 partitions shown)

This administrative utility is provided as a sample only and is not to be considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:9809

arning: Detected there is more than one Placement Service MBean

*****Displaying Results for Grid - DYNACACHE_REMOTE, MapSet - DYNACACHE_REMOTE*****

*** Listing Maps for thinkCell01\thinkNode02\WXS_DC_ClusterMember2 ***

Map Name	Partition	Map Size	Used Bytes (KB)	Shard Type
IBM_DC_PARTITIONED_baseCache 0	17	21		Primary
IBM_DC_PARTITIONED_baseCache 1	19	23		AsynchronousReplica
IBM_DC_PARTITIONED_baseCache 2	10	11		AsynchronousReplica
IBM_DC_PARTITIONED_baseCache 3	19	24		AsynchronousReplica
IBM_DC_PARTITIONED_baseCache 4	14	17		AsynchronousReplica
...				

Server Total: 618 (761KB)

*** Listing Maps for thinkCell01\thinkNode02\WXS_DC_ClusterMember1 ***

Map Name	Partition	Map Size	Used Bytes (KB)	Shard Type
IBM_DC_PARTITIONED_baseCache 0	17	21		AsynchronousReplica
IBM_DC_PARTITIONED_baseCache 1	12	14		Primary
IBM_DC_PARTITIONED_baseCache 2	10	11		Primary
IBM_DC_PARTITIONED_baseCache 3	8	9		Primary
IBM_DC_PARTITIONED_baseCache 4	12	14		Primary
...				

Server Total: 648 (801KB)

Total Domain Count: 1266 (1.53MB)

For information about using **xsadmin**, see 3.8.2, “Monitoring with xsadmin” on page 49.

6.4.4 Further monitoring options

There are additional tools available that provide the same or more detailed information to the tools we have listed so far. This section gives an overview of those tools.

The WebSphere eXtreme Scale web console

The WebSphere eXtreme Scale web console is a new feature with version 7.1. It is primarily used for sizing and performance related information. At the time of this writing, the web console is not installed when you install eXtreme Scale on WebSphere Application Server. Rather, it is only installed with a stand-alone version. The reason for this is that the data is typically accessible through the Performance Viewer in the administrative console.

However, if you want to use the web console with WebSphere Commerce, simply perform a stand-alone installation of eXtreme Scale and you can use the console. You can use the copy in the stand-alone installation to monitor an eXtreme Scale grid hosted on WebSphere Application Server or a stand-alone grid.

For information about using the web console, see 3.8.1, “Using the eXtreme Scale web console” on page 47.

MBean statistics using JConsole

The MBean statistics in JConsole provide extensive and detailed metrics available for eXtreme Scale grids, containers, and shards. See 3.8.5, “Monitoring MBean statistics in JConsole” on page 55 for more information.

Cache statistics logging

The Cache Monitor application can only provide data when requested. Sometimes it is useful to log cache statistics over a period of time. A wsadmin script is available from WebSphere development that can log the data in CSV format, which can then be exported to a spreadsheet for further analysis. The script collects cache statistics exposed by the Dynacache mbean at a periodic interval.

This can be run in production environments over time, with minimal impact. For further information, go to the following address:

<http://wasdynacache.blogspot.com/2010/03/dynacache-flight-recorder-wsadmin.html>

6.5 Sizing guidance

The final design consideration for WebSphere eXtreme Scale dynamic cache provider is how large the eXtreme Scale grid should be. This depends on your current environment. If you already use the default dynamic cache provider, then you probably know how much data you cache today, and therefore how much you plan to cache with eXtreme Scale. You will currently have your environment set to either a maximum number of cache entries or maximum cache size. Also, the Cache Monitor application can show you how well utilized the cache is, which will also give a good indication of how many objects it is desirable to cache.

Use the following metrics to determining how to size an eXtreme Scale grid. You might need to vary the calculations based on your environment, but the principles apply broadly. Determine the following factors:

- ▶ How many cache entries you want to store? In our example, we use 100,000 entries.
- ▶ What is the average size of a data entry is going to be in eXtreme Scale? Estimate the size using the Extended Cache Monitor MBean statistics.

In eXtreme Scale, this data will be compressed. For web content, estimate a compression ratio of 0.4, which is the worse compression than you're likely to see.

Avg object size=
 $\text{MemoryCacheSizeInMB} * \text{Compression Ratio} / \text{MemoryCacheEntries}$

In our example, the average cached object size is 25kb.

If you do not yet have these metrics for your Commerce site, take a typical page and compress it using an archive tool to give an indicative size.

- ▶ How resilient do you want your cache? In eXtreme Scale terms this is how many replicas you want. The default of one asynchronous replica will likely be sufficient.
- ▶ How much space do you have in a WebSphere eXtreme Scale container or JVM?
Do not exceed 70% of your application server heap to allow room for garbage collection. Also, leave room for JVM and WebSphere services to run. You can verify how much space would be available by looking at how much memory an empty WebSphere Application Server takes up. In our test, it was around 50Mb.

For our example, we will assume that out of a maximum heap of 1.5 Gb, we have 1 Gb for caching data.

Having collated or estimated the prerequisite data, size your grid as follows:

1. Calculate the total size of the cached data.

Total cache size = Avg object size * Num entries to cache * (1 + Num replicas)

For our example, the total cached data would be:

$25\text{kb} * 100000 * (1 + 1) = 5000000\text{kb}$ or 5Gb

2. Calculate how many eXtreme Scale containers are needed to host the data.

Num eXtreme Scale containers = Total cache size / Free memory per container

For our example, the number of containers would be:

$5 / 1 = 5$ containers

Important: Consider your availability requirements in deciding the number of eXtreme Scale containers. If the number of containers comes out quite low, then determine the number of containers based on what minimum is required to have an available configuration. For example, three or four containers spread over two servers is the recommended minimum to ensure an even spread of data partitions in the grid.

3. Work out the cache size setting.

Remember that the cache size setting on the application server dynamic cache service relates to separate things depending on whether you are using eXtreme Scale as the dynamic cache provider. By default, the cache size setting refers to the size of the whole cache. With eXtreme Scale, the cache size setting refers to the number of cache entries *per partition*.

Cache size setting = Num entries to cache / Num partitions

For our example, the cache size setting would be:

$100000 / 47 = 2128$ cache entries per partition

Tip: The number partitions is also configurable. The default of 47 is adequate for small to medium dynamic cache grids because it gives an even balance of data and load across the grid. Increase the number of partitions for a larger grid. There is no absolute guidance for this, but to continue to have an even balance of data and load on the grid, aim to have at least five partitions per container.

For an alternative approach to sizing, where you need to deduce how many cache entries you can store in an existing or known eXtreme Scale grid capacity, see the Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r0/index.jsp?topic=/com.ibm.webSphere.extremescale.admin.doc/cxscapacityplan.html>

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *User's Guide to WebSphere eXtreme Scale*, SG24-7683
- ▶ *IBM WebSphere eXtreme Scale V7: Solutions Architecture*, REDP-4602
- ▶ *Mastering DynaCache in WebSphere Commerce*, SG24-7393
- ▶ *WebSphere Application Server V7 Administration and Configuration Guide*, SG24-7615

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, and order hardcopy Redbooks publications, at this website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ IBM Support Fix Central
<http://www-933.ibm.com/support/fixcentral/>
- ▶ WebSphere eXtreme Scale Version 7.1 Cumulative Fix 1
<http://www-01.ibm.com/support/docview.wss?uid=swg27018991#ver71>
- ▶ Upgrading and migrating WebSphere eXtreme Scale Version 7.1
<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.admin.doc/txsupdate.html>
- ▶ L2 cache Plugin Support
<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r0/index.jsp?topic=/com.ibm.websphere.extremescale.admin.doc/cxscacheint.html>
- ▶ Catalog server quorums
<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r0/topic/com.ibm.websphere.extremescale.over.doc/cxsquorcatsr.html>
- ▶ WebSphere eXtreme Scale Availability
<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r0/topic/com.ibm.websphere.extremescale.over.doc/cxsavail.html>
- ▶ Monitoring with vendor tools
<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.admin.doc/cxsmonitorvendor.html>

- ▶ Deprecated APIs

http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.wpf.doc/cwpfpartition_pdf.html

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.javadoc.doc/topics/com/ibm/websphere/objectgrid/streamquery/package-summary.html>

- ▶ Client for WebSphere eXtreme Scale Version 7.1 download

<http://www-01.ibm.com/support/docview.wss?uid=swg24028267>

- ▶ Starting a stand-alone catalog service

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.admin.doc/txscatalogstart.html>

- ▶ **xsadmin** utility

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.admin.doc/txsxsadmin.html>

- ▶ WebSphere Real Time Java

<http://www-01.ibm.com/software/webservers/realtime>

- ▶ WebSphere eXtreme Scale Information Center

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.over.doc/cxsdynacache.html>

- ▶ Enhancing WebSphere Commerce performance with WebSphere eXtreme Scale

http://www.ibm.com/developerworks/websphere/techjournal/1008_genkin/1008_genkin.html

- ▶ Dynamic cache engine and eXtreme Scale functional differences

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.over.doc/cxsdynacache.html>

- ▶ Configuring ports in stand-alone mode

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/topic/com.ibm.websphere.extremescale.admin.doc/txscfgtcp.html>

- ▶ Starting and stopping stand-alone servers

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.admin.doc/txssastartstop.html>

- ▶ Dynamic cache service settings

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/udyn_rcachesettings.html

- ▶ Configure and manage the dynamic cache service settings

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/udyn_rcachesettings.html

- ▶ WebSphere Commerce data cache

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.admin.doc/tasks/tdccdcont.htm>

- ▶ IBM Extended Cache Monitor for IBM WebSphere Application Server

http://www.ibm.com/developerworks/websphere/downloads/cache_monitor.html

- ▶ Displaying cache information

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tdyn_servletmonitor.html

- Monitoring performance with Tivoli Performance Viewer (TPV)

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tprf_tpvmonitor.html

- Monitoring with the web console

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1/index.jsp?topic=/com.ibm.websphere.extremescale.admin.doc/txsmonitoroversw.html>

- Dynacache Blog

<http://wasdynacache.blogspot.com/2010/03/dynacache-flight-recorder-wsadmin.html>

- WebSphere eXtreme Scale capacity planning and high availability

<http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r0/index.jsp?topic=/com.ibm.websphere.extremescale.admin.doc/cxscapacityplan.html>

- Integrating WebSphere Business Events with WebSphere eXtreme Scale

<http://publib.boulder.ibm.com/infocenter/wbevents/v7r0m1/topic/com.ibm.wbe.integrating.doc/doc/integrating-wxs.html>

- Jazz community site

<http://jazz.net>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redbooks®

Scalable, Integrated Solutions for Elastic Caching Using IBM WebSphere eXtreme Scale

**Easily scalable
solutions for elastic
data caching**

**Integration with IBM
WebSphere
Commerce Server**

**Integration with IBM
WebSphere Portal
Server**

IBM WebSphere eXtreme Scale provides a powerful, elastic, high-performance solution for scalability issues through caching and grid technology. This IBM Redbooks publication shows architects and IT personnel how to leverage the power of WebSphere eXtreme Scale technology to enhance data caching performance in their enterprise networks.

This book discusses the scalability challenges and solutions facing today's dynamic business and IT environments. Topics discussed include existing scalability solutions, how WebSphere eXtreme Scale can be integrated into these solutions, and best practices for using WebSphere eXtreme Scale in different environments, including application data caching and database caching. Also included is an in-depth discussion of the WebSphere eXtreme Scale infrastructure, such as grid clients and servers, the grid catalog service, zone support, and scalability sizing considerations.

This book focuses on the challenges and benefits of integrating WebSphere eXtreme Scale with other middleware products, including WebSphere Business Events, WebSphere Commerce, WebSphere Portal, and Rational Jazz-based products. Detailed procedures for integrating, configuring, and monitoring WebSphere eXtreme Scale in WebSphere Portal and WebSphere Commerce environments are provided.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-7926-00

ISBN 0738435201